UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SYRU/DECE/TR-84/3	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOLUTION OF SOME ADDITIONAL ELECTROMAGNETIC PROBLEMS BY THE DISCRETE CONVOLUTION METHOD		5. TYPE OF REPORT & PERIOD COVERED Technical Report No. 23 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Htay L. Nyo Roger F. Harrington		8. CONTRACT OR GRANT NUMBER(s) N00014-76-C-0225
9. PERFORMING ORGANIZATION NAME AND ADDRESS Dept. of Electrical & Computer Engineering Syracuse University Syracuse, New York 13210		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE February 1984 13. NUMBER OF PAGES 77
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer Program
Discrete Convolution
Large Matrices
Method of Moments

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report gives solution results and computer programs for the Discrete Convolution Method applied to scattering from a helix, to radiation from planar array antennas with antenna elements arranged in triangular patterns (solved using one expansion function per element), and to radiation from planar array antennas with antenna elements arranged in triangular patterns (solved using three expansion functions per element). It also gives a brief discussion on the spatial domain interpretation of the spectral domain iteration technique.

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LIBRARY
RESEARCH REPORTS DIVISION
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

SYRU/DECE/TR-84/3

SOLUTION OF SOME ADDITIONAL ELECTROMAGNETIC PROBLEMS BY THE DISCRETE CONVOLUTION METHOD

by

Htay L. Nyo
Roger F. Harrington

Department of
Electrical and Computer Engineering
Syracuse University
Syracuse, New York 13210

Technical Report No. 23
February 1984

Contract No. N00014-76-C-0225

Approved for public release; distribution unlimited

Reproduction in whole or in part permitted for any purpose of the United States Government.

Prepared for

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
ARLINGTON, VIRGINIA 22217

SYRU/DECE/TR-84/3

SOLUTION OF SOME ADDITIONAL ELECTROMAGNETIC PROBLEMS BY THE DISCRETE CONVOLUTION METHOD

by

Htay L. Nyo
Roger F. Harrington

Department of
Electrical and Computer Engineering
Syracuse University
Syracuse, New York 13210

Technical Report No. 23
February 1984

Contract No. N00014-76-C-0225

Approved for public release; distribution unlimited

Reproduction in whole or in part permitted for any purpose of the United States Government.

Prepared for

DEPARTMENT OF THE NAVY
OFFICE OF NAVAL RESEARCH
ARLINGTON, VIRGINIA 22217

CONTENTS

I. INTRODUCTION

In the first report [1] on discrete convolution method for solving some large moment equations, three types of one and two dimensional problems were solved. In this report, we solve three more types of one and two dimensional problems. They are

(i) scattering from a helix

(ii) planar arrays with antenna elements arranged in triangular pattern, solved using one expansion function per element

(iii) planar arrays with antenna elements arranged in triangular pattern, solved using three expansion functions per element

As in the first report, the computing time measurements (made on a KL/10 machine) and the number of iterations needed for the given accuracy are listed. In addition, the array factor of the planar arrays are given.

II. SAMPLE COMPUTATIONS AND COMMENTS

The "one" dimensional problem is scattering from a helix. Fig. 1 shows the problem of scattering from a helix. The MOM formulation of the helix problem requires that the helix be subsectioned into equal length segments. If we number the helix segments so that the segment numbers are in consecutive increasing order from top to bottom, then it is

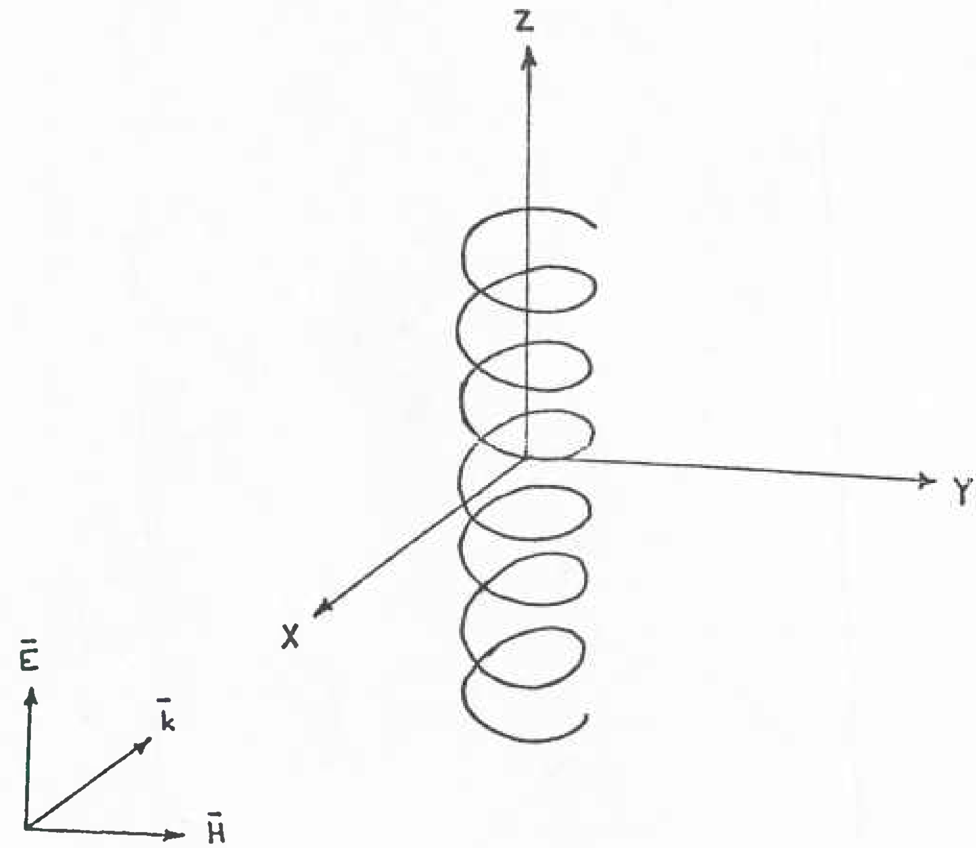


Fig. 1. The helical wire scatterer

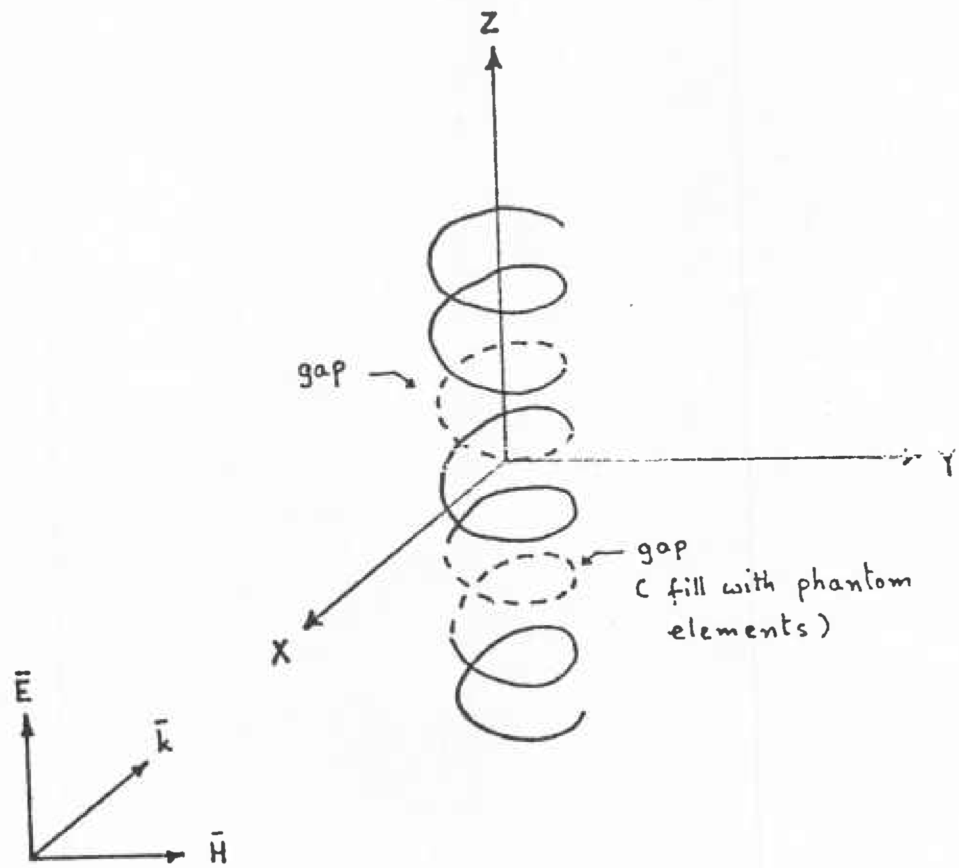


Fig. 2. The helical wire scatterer with gaps

easy to see that the mutual coupling between segments as given by the MOM impedance matrix Z is

$$Z_{mn} = Z(m-n) \qquad (1)$$

i.e., the value is dependent only on the difference between segment numbers. Therefore the Z matrix will be Toeplitz if the helix is a complete helix as shown in Fig. 1. The Z matrix will be non-Toeplitz if the helix has gaps in between as shown in Fig. 2. It is apparent from the discussions of other one dimensional problems that both problems can be solved using the one dimensional DCM technique. For the helix with gaps all we need is to insert phantom elements as shown in Fig. 2.

Table 1 gives the number of iterations needed to get the required degree of accuracy for the helix problem. We can see that the number of iterations needed is practically independent of the length of the helix.

The problem of a planar array with antennas arranged in triangular patterns instead of rectangular, can also be formulated as a two dimensional convolution equation by adding phantom elements (as shown in Fig. 4), to make a parallelogram. The triangular pattern arrangement is shown in Fig. 3.

The MOM formulation using one expansion per antenna then gives a block Toeplitz matrix which can be solved using two dimensional DCM. However, it cannot be solved using the block Toeplitz method since the field on the phantom

Table 1. Results for some helical scatterer problems

Radius	Pitch	Number of turns	Number of segments	Number of iterations	Field error (%)	Last current change (%)
.125	.30	6	120	22	.693	.823
					.0498	.1
.125	.30	6	120	23*	1.01	1.11
					.141	.113
.15	.50	6	120	8	.812	1.39
					.0504	.202
				11	.0562	.0958
					.0036	.0142
.15	.30	6	120	10	1.56	1.17
					.177	.248
				16	.0668	.0496
					.0076	.0109
.15	.30	8	160	12	.568	1.21
					.0375	.0778
				16	.0697	.153
					.0046	.0097
.15	.30	8	160	13*	.453	.883
					.0572	.0467
				16*	.0934	.184
					.0118	.0097

Table 1. continued

Radius	Pitch	Number of turns	Number of segments	Number of iterations	Field error (%)	Last current change (%)
.15	.30	8	160	11*	1.07	.785
					.104	.141
				16*	.0821	.0602
					.0082	.0114
.15	.30	16	320	13	.395	.659
					.0154	.0295
				16	.0848	.139
					.0033	.00634
.25	.50	8	160	11	.537	1.86
					.0282	.2
				16	.0495	.163
					.0026	.0184

Here, Radius is the radius of the helix in wavelengths

Pitch is the pitch of the helix in wavelengths

the wire radius in all cases is .006 wavelengths

first entries for each problem in the last two columns are maximum values and second entries are average values

# indicates that polarity of impressed E field is not as shown in Fig. 10 but is z directed

* indicates that the impressed field is not from the direction shown in Fig. 10 but is from z direction

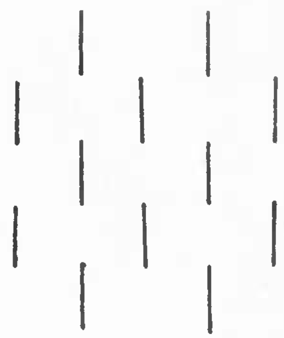


Fig. 3. The triangular pattern arrangement.

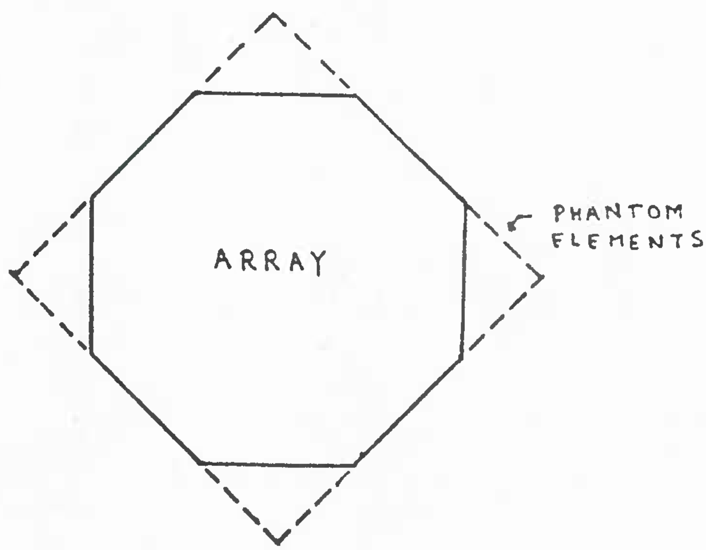


Fig. 4. A planar array with antennas arranged in triangular pattern.

elements are unknown. Therefore, only LU decomposition or two dimensional DCM can be used. For a large array, DCM will be considerably faster. Since the current on each antenna is not symmetric, using three expansion functions per antenna gives a much more accurate result and needs five times more computing time for the DCM. With LU decomposition method computing time will go up twenty seven times the already large value.

Table 2 lists the computing time and number of iterations needed for the DCM solution using one expansion function per antenna element. Table 3 lists the computing time and number of iterations needed for the DCM solution using three expansion functions per antenna element. All the problems are for planar arrays with 0.48 wavelength antennas one-quarter wavelength in front of the infinite ground plane. The seperation between antennas is one-half wavelength in either direction.

The graphs given in Figs. 6, 7, 8, 9, 10, and 11 are the arrays factors for the planar arrays with triangular pattern arrangement solved by the DCM using one expansion function per antenna element. The array factors are computed in the plane perpendicularly bisecting the array as shown in Fig. 5. Angle measurements are as shown. In all the figures, the solid lines give the array factors for the solutions which take the mutual coupling between antennas into account and the dashed lines are for the idealized solutions which do not take the mutual coupling into

Table 2. Results for some planar arrays with triangular pattern arrangement

N	Excitation	I	Computing Time(secs)	Field Error (%)	Current Change(%)
12	Uniform	5	4	.1789	.664
				.0867	.378
	Beam steer	6	4	.1124	.595
	(45°)			.0554	.173
76	Uniform	6	20	.1447	.489
				.0283	.116
	Beam steer	6	20	.2768	.796
	(45°)			.0333	.098
372	Uniform	5	105	.4950	1.674
				.0467	.174
	Beam steer	6	105	.3636	.863
	(45°)			.0163	.0456

Here, N is the number of antennas in the array

I is the number of iterations needed to get the given accuracy. For both field error and (last) current change, the upper entry is the maximum and the lower entry is the average.

Table 3. Results for some planar arrays with triangular pattern arrangement (Multiple Expansion Solutions)

N	Excitation	I	Computing Time (secs)	Field Error (%)	Current Change (%)
76	Uniform	6	67	.0401	.3612
				.0027	.0836
	Beam steer	6	67	.078	.6898
	(xz 45°)			.0032	.0704
	Beam steer	6	67	.0251	.527
	(yz 135°)			.0016	.0493
372	Uniform	5	165	.1450	1.298
				.0048	.134
	Beam steer	5	165	.2928	2.1100
	(xz 45°)			.0047	.0994
	Beam steer	5	165	.0706	1.43
	(yz 135°)			.0029	.0748

Here, N is the number of antennas in the array

I is the number of iterations needed to get the given accuracy. For both field error and (last) current change, the upper entry is the maximum and the lower entry is the average.

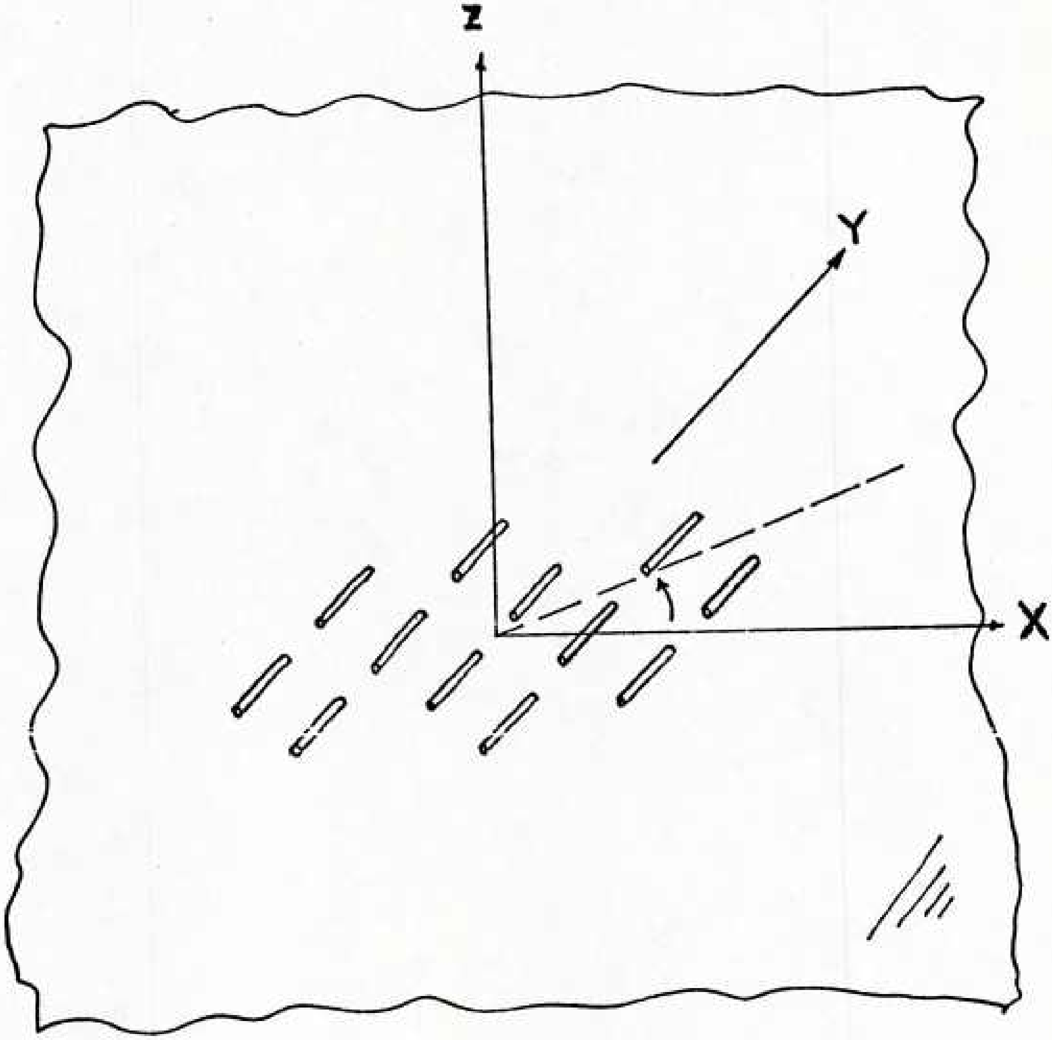


Fig. 5. The relative position of the plane in which the array factors are computed.

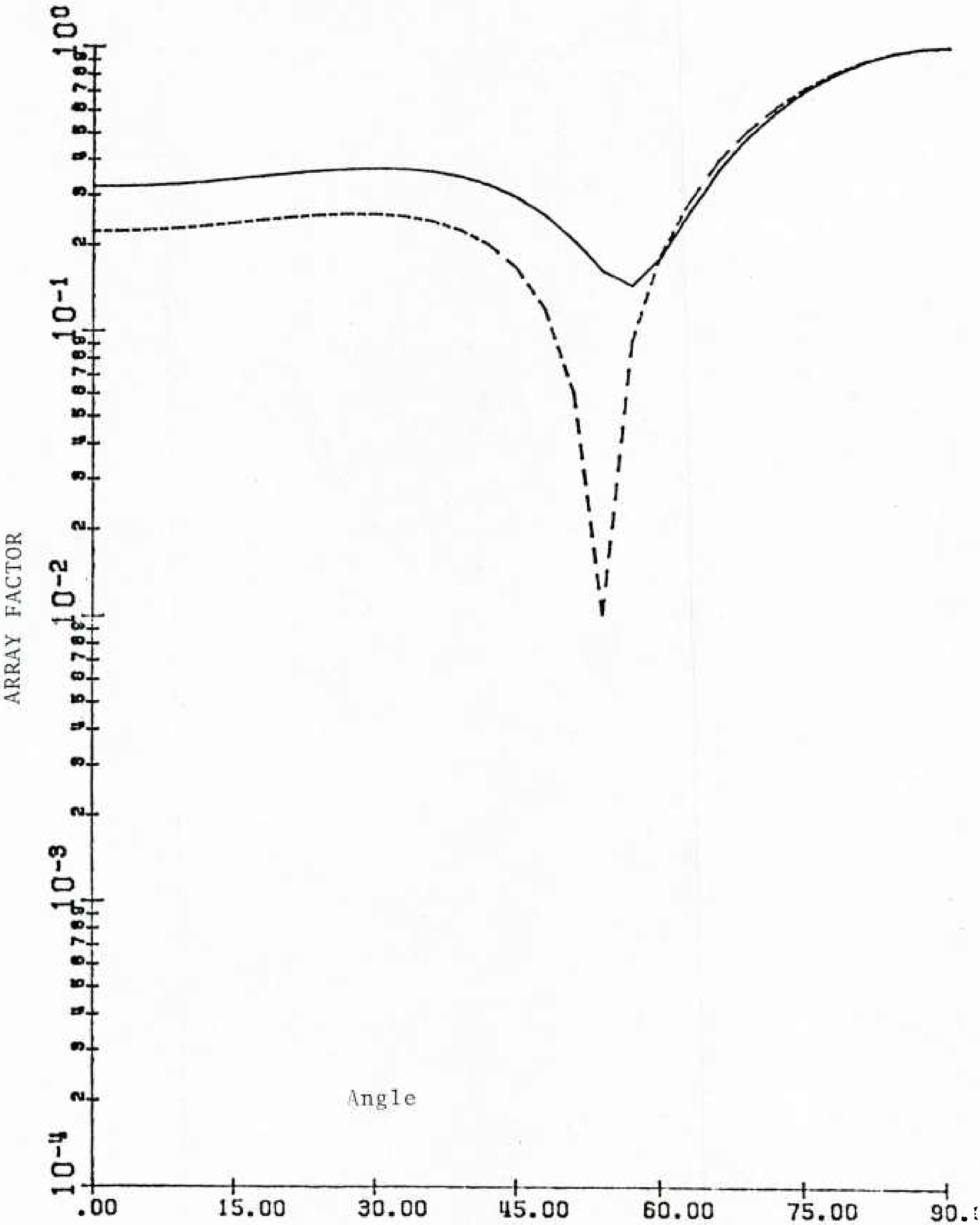


Fig. 6 Array factor of the twelve antenna planar array with uniform excitation

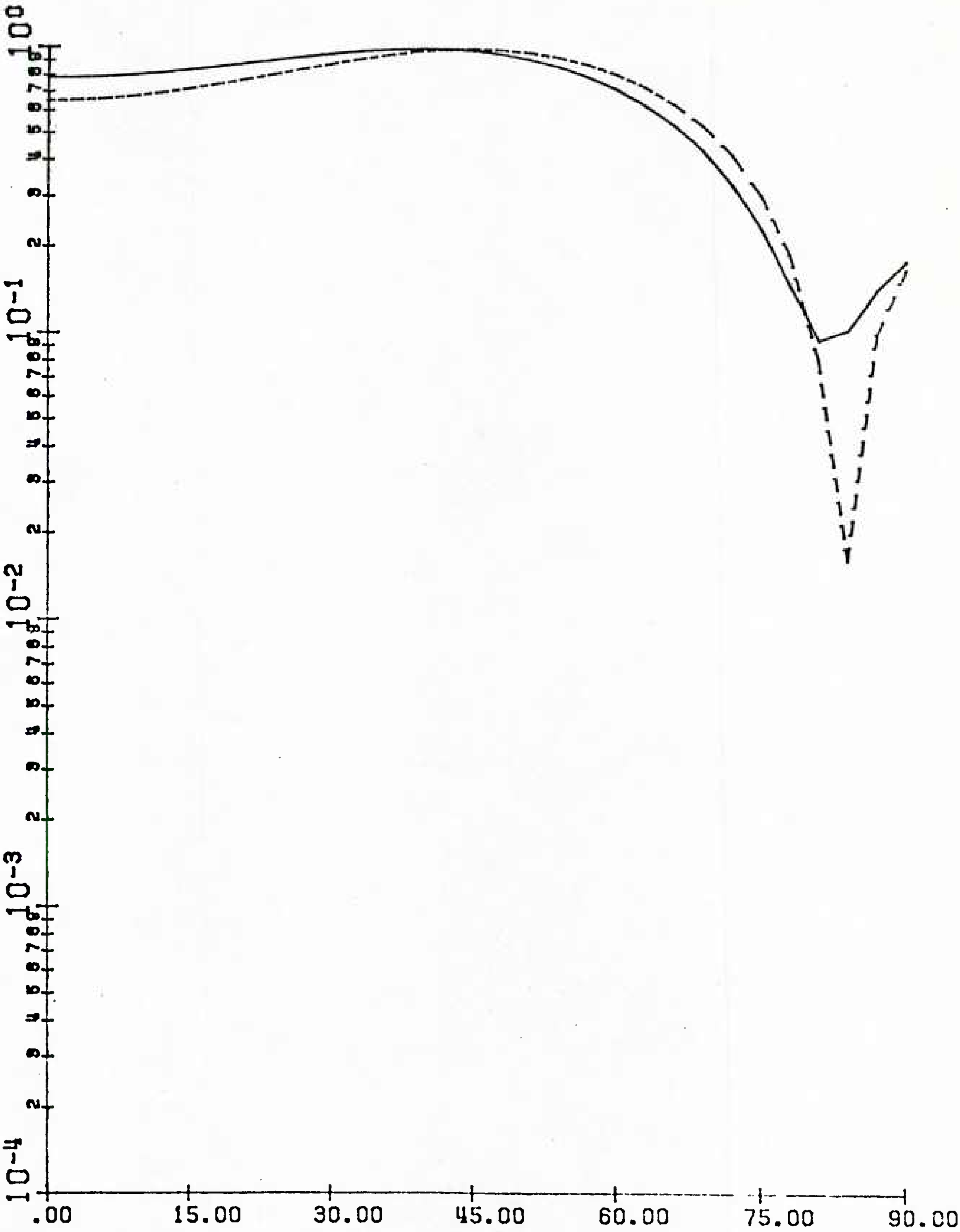


Fig. 7 Array factor of the twelve antenna planar array with excitation to give a 45 degrees scan

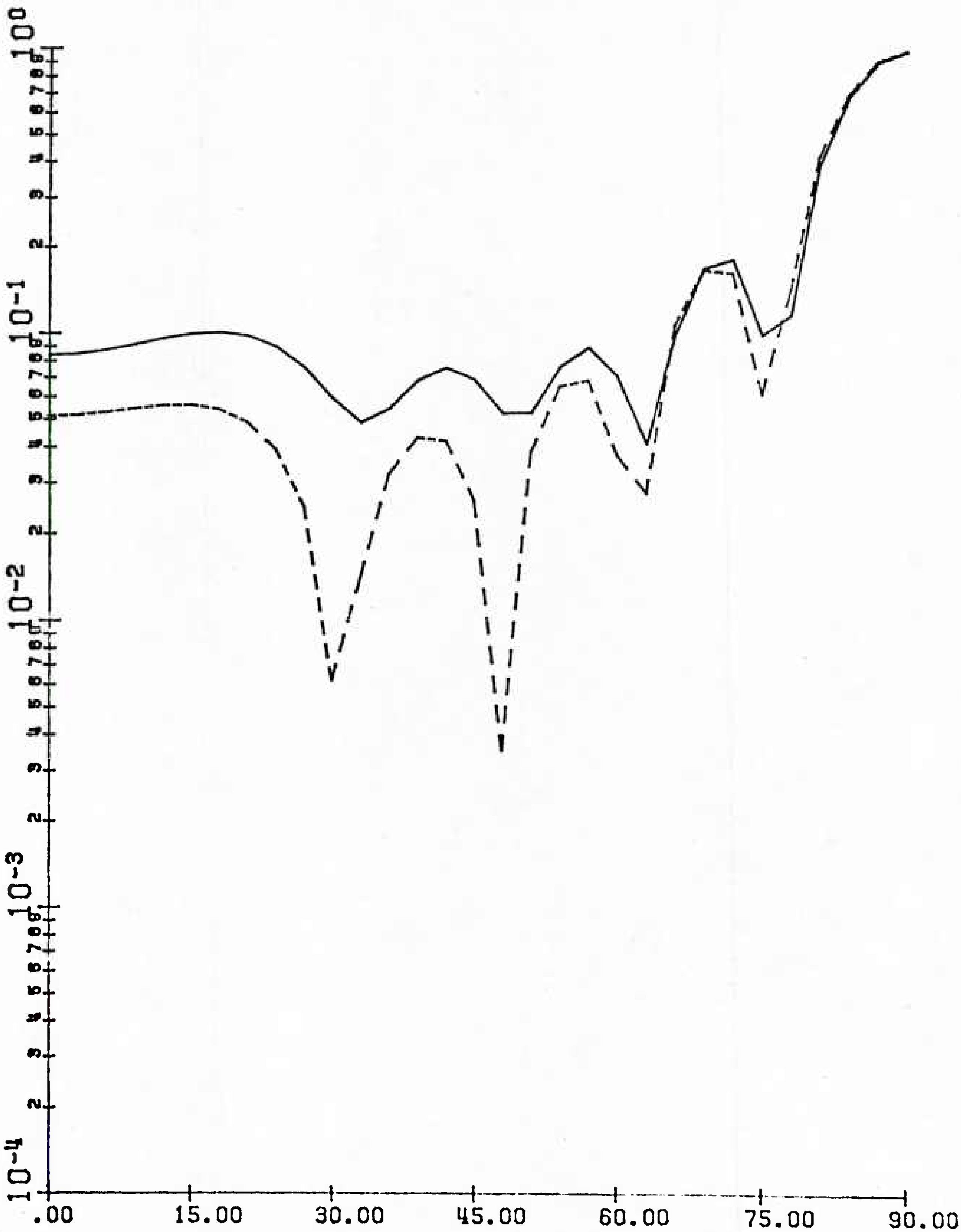


Fig. 8 Array factor of the seventy six antenna planar array with uniform excitation

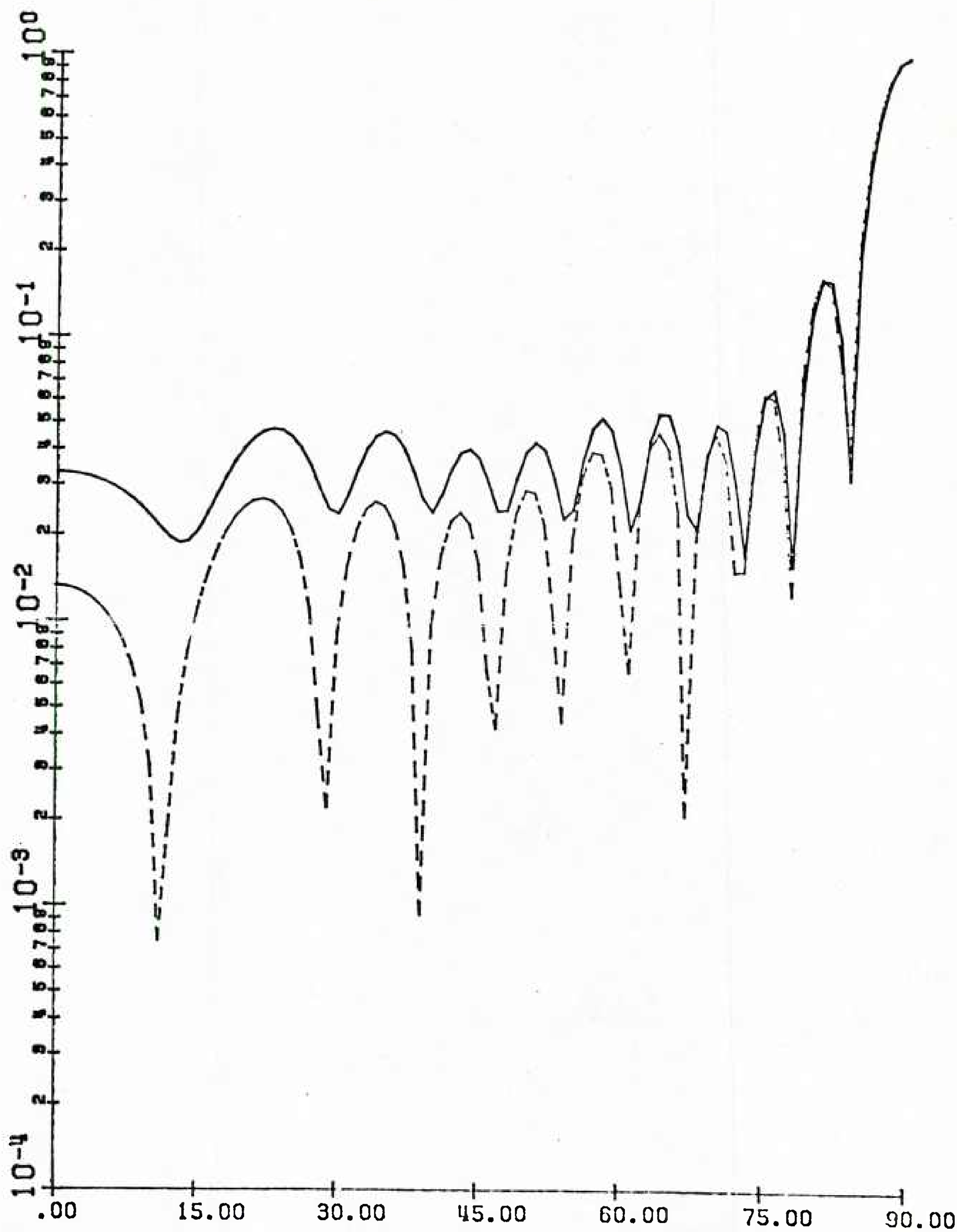


Fig. 10 Array factor of the three hundred and seventy two antenna planar array with uniform excitation

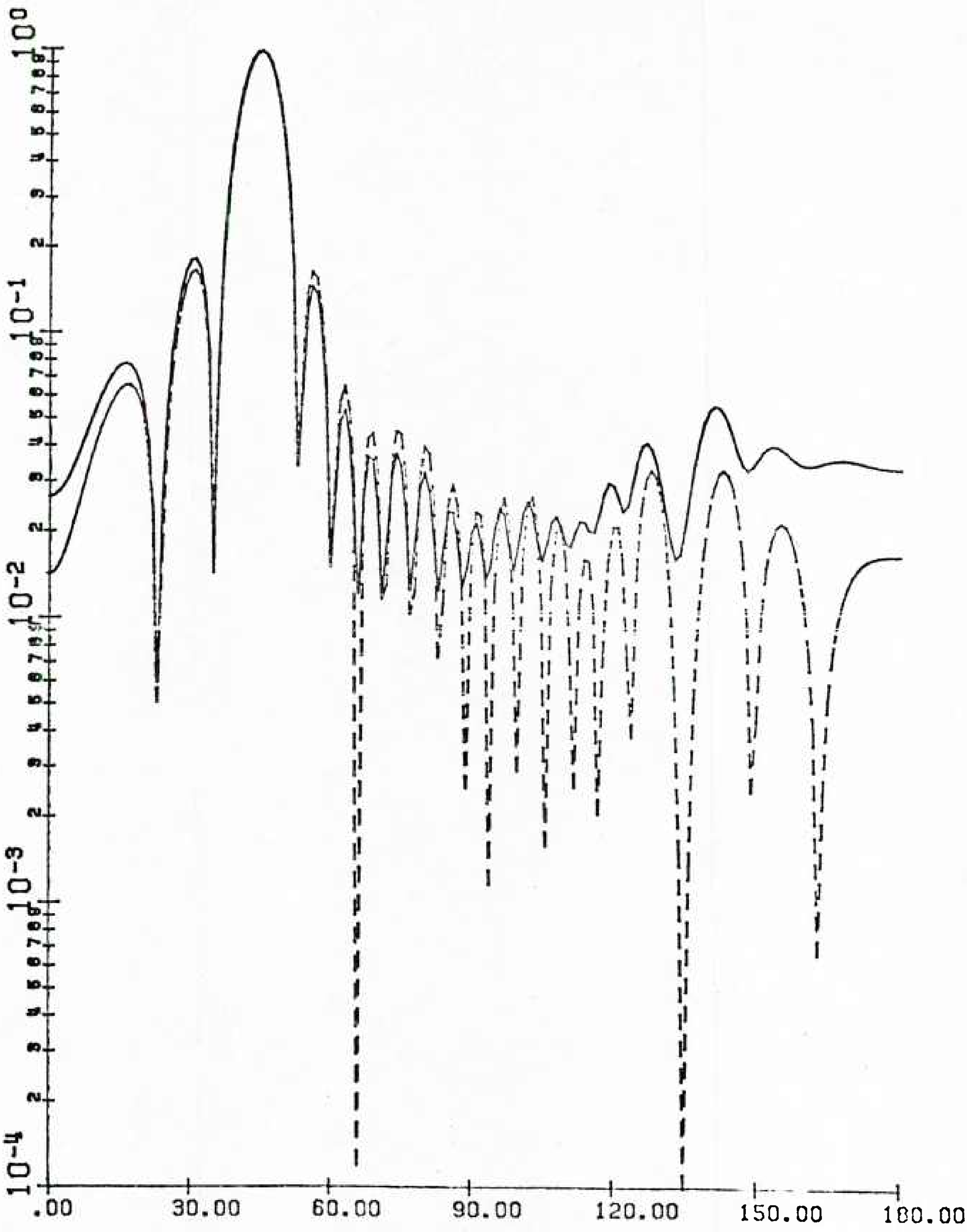


Fig. 11 Array factor of the three hundred and seventy two antenna planar array with excitation to give a 45 degrees scan

account.

As we can see from the graphs, for larger arrays, the main beam is not effected by the mutual coupling but the side lobes and the nulls are effected strongly.

The graphs given in Figs. 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, and 24 are the arrays factors for the planar arrays with triangular pattern arrangement solved by the DCM using three expansion functions per antenna element. The array factors are computed in the xz and yz planes perpendicularly bisecting the array as shown in Fig. 12. Angle measurements are as shown. In all the figures, the solid lines give the array factors for the solutions which takes the mutual coupling between antennas into account and the dashed lines are for the idealized solutions which do not take the mutual coupling into account.

The array factor is defined as the far field pattern divided by the element factor. Therefore for the DCM solution using one expansion function per element, the element factor is the far field pattern of the expansion function and so the array factor can be and is computed directly from the solved currents. However, with three expansions per antenna element, the current distribution over each element is different from the others. Therefore, since our main purpose in computing array factors is to compare the far field patterns, we define a normalized array factor as the total far field pattern divided by the far

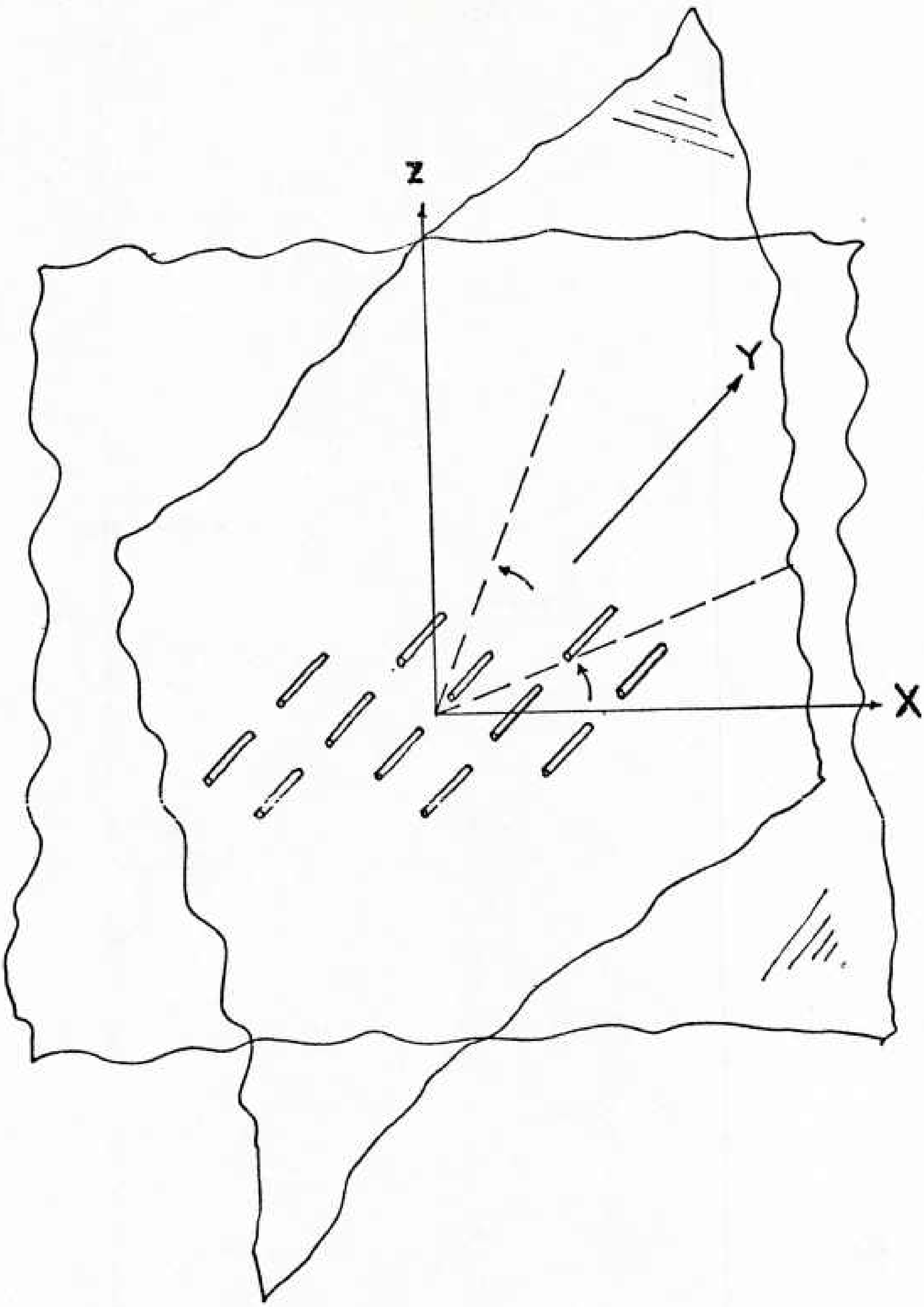


Fig. 12. The relative position of the planes in which the array factors are computed.

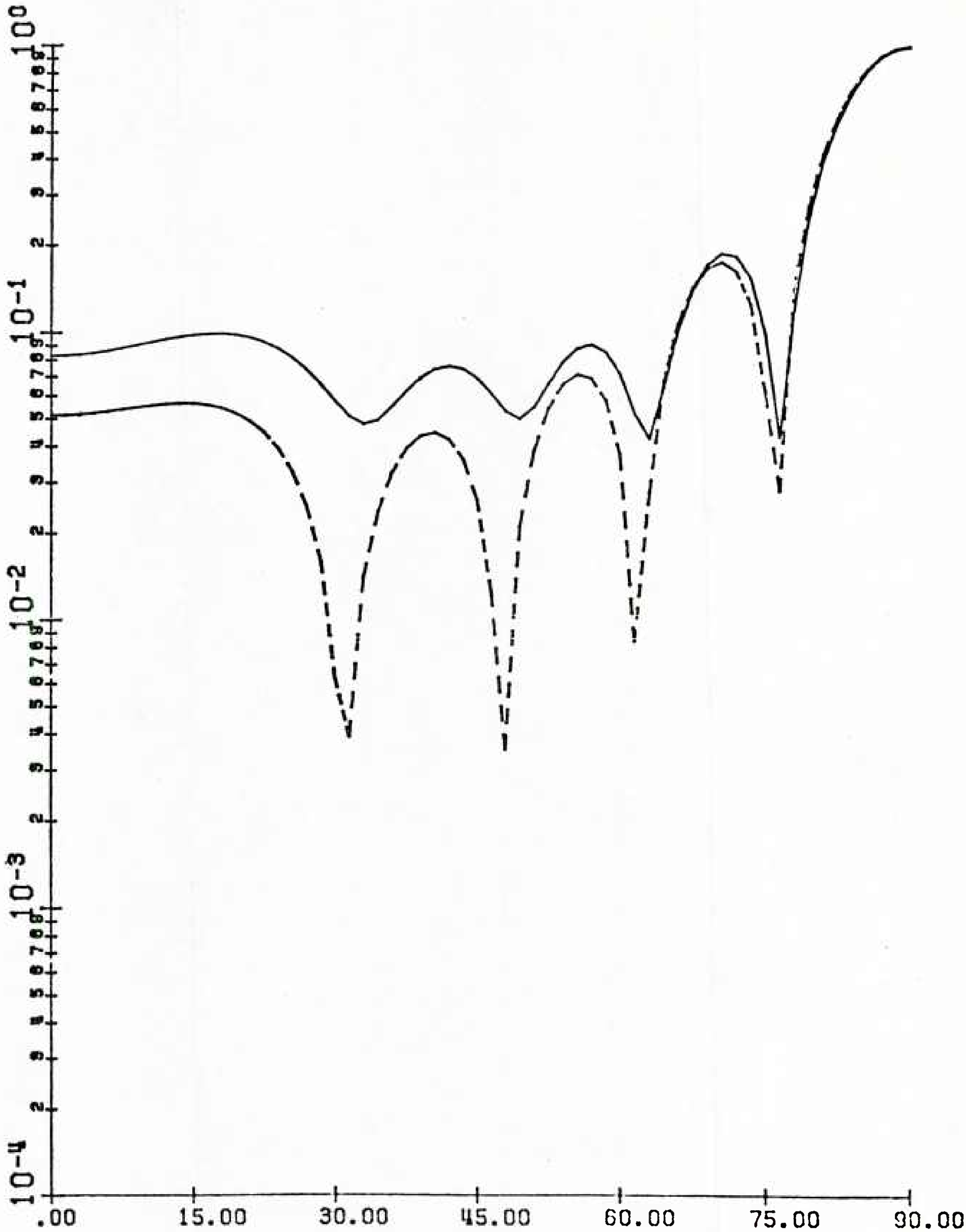


Fig. 13 Array factor in the xz plane of the seventy six antenna planar array with uniform excitation

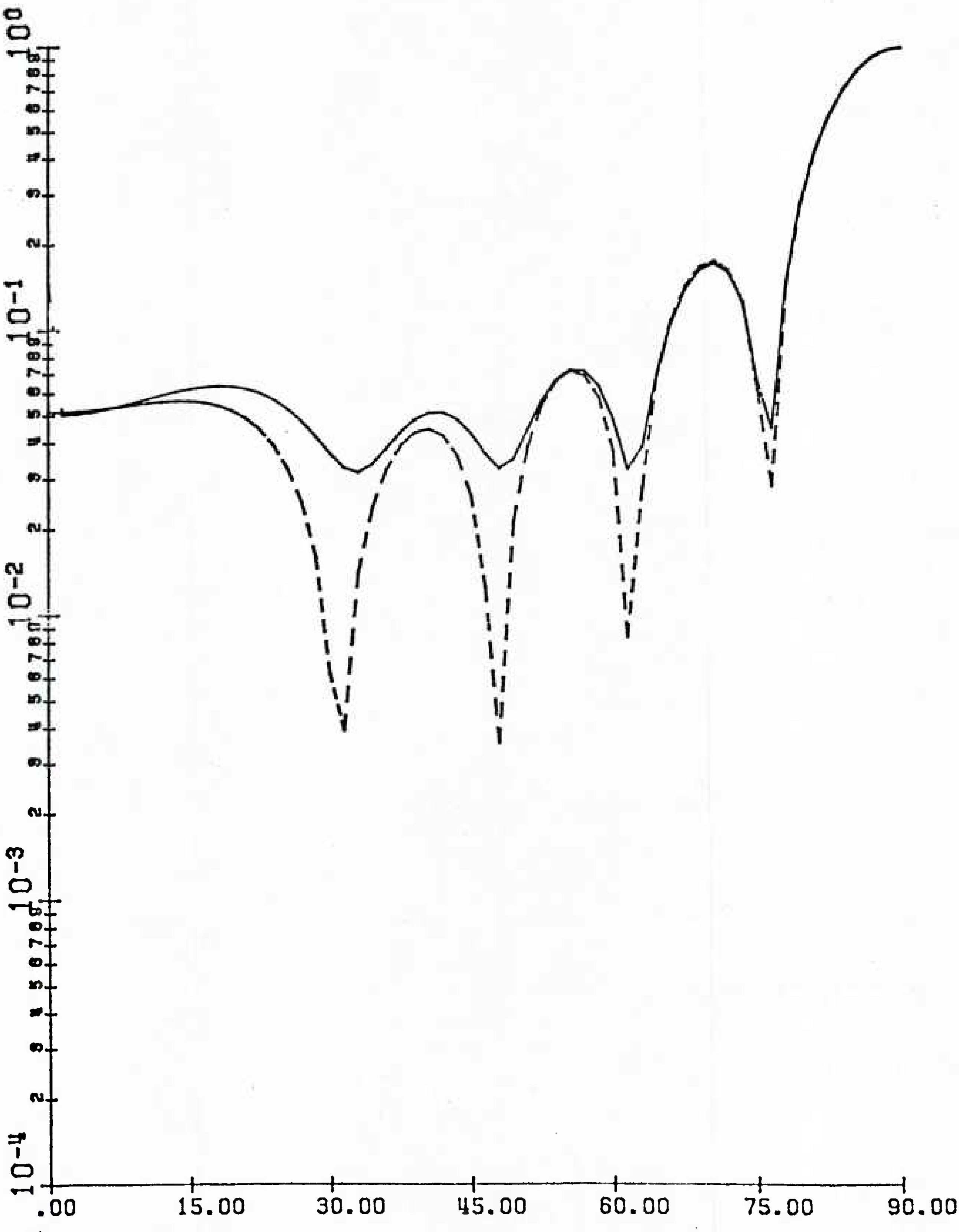


Fig. 14 Array factor in the yz plane of the seventy six antenna planar array with uniform excitation

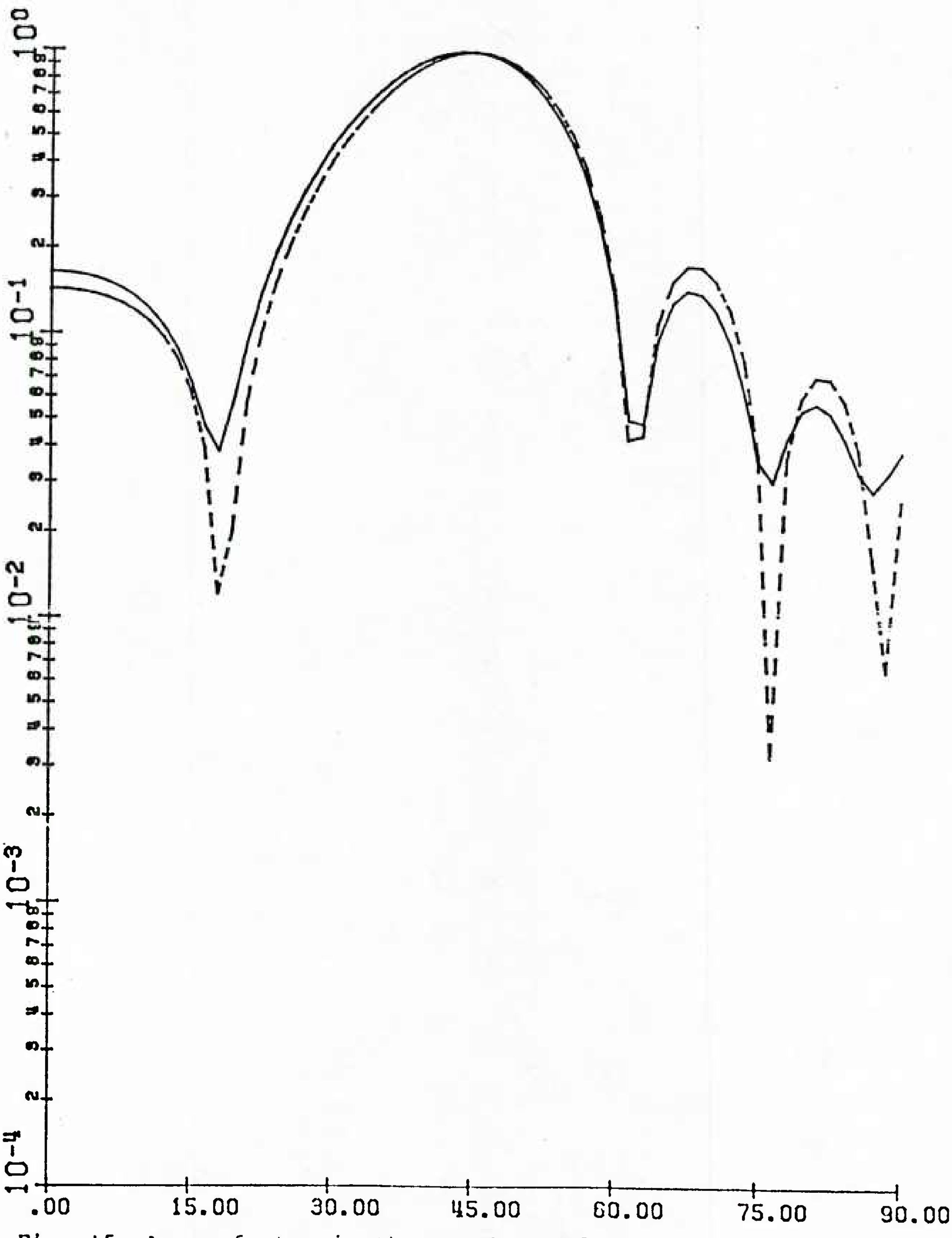


Fig. 15 Array factor in the xz plane of the seventy six antenna planar array with excitation to give a 45 degrees scan in the xz plane

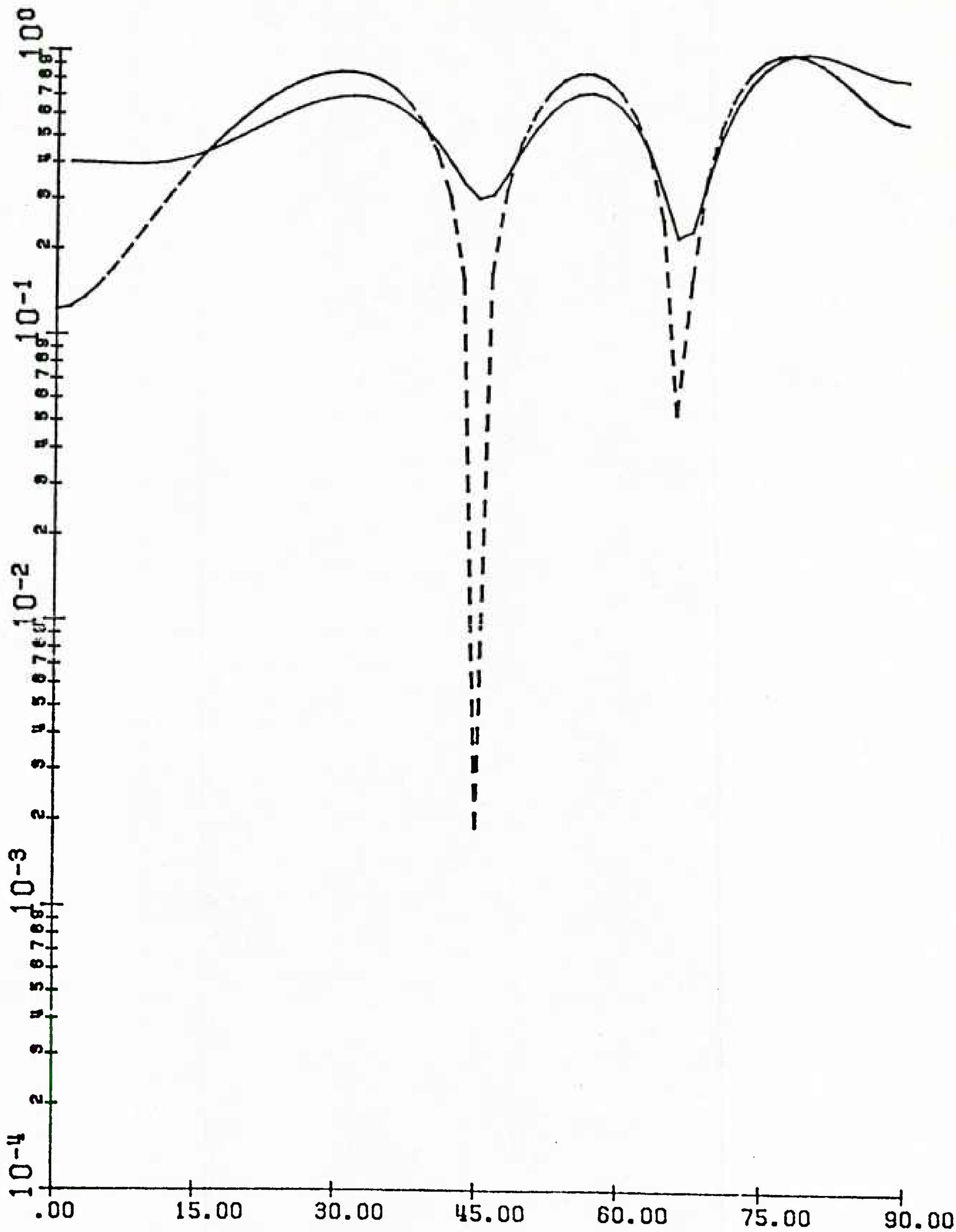


Fig. 16 Array factor in the yz plane of the seventy six antenna planar array with excitation to give a 45 degrees scan in the xz plane

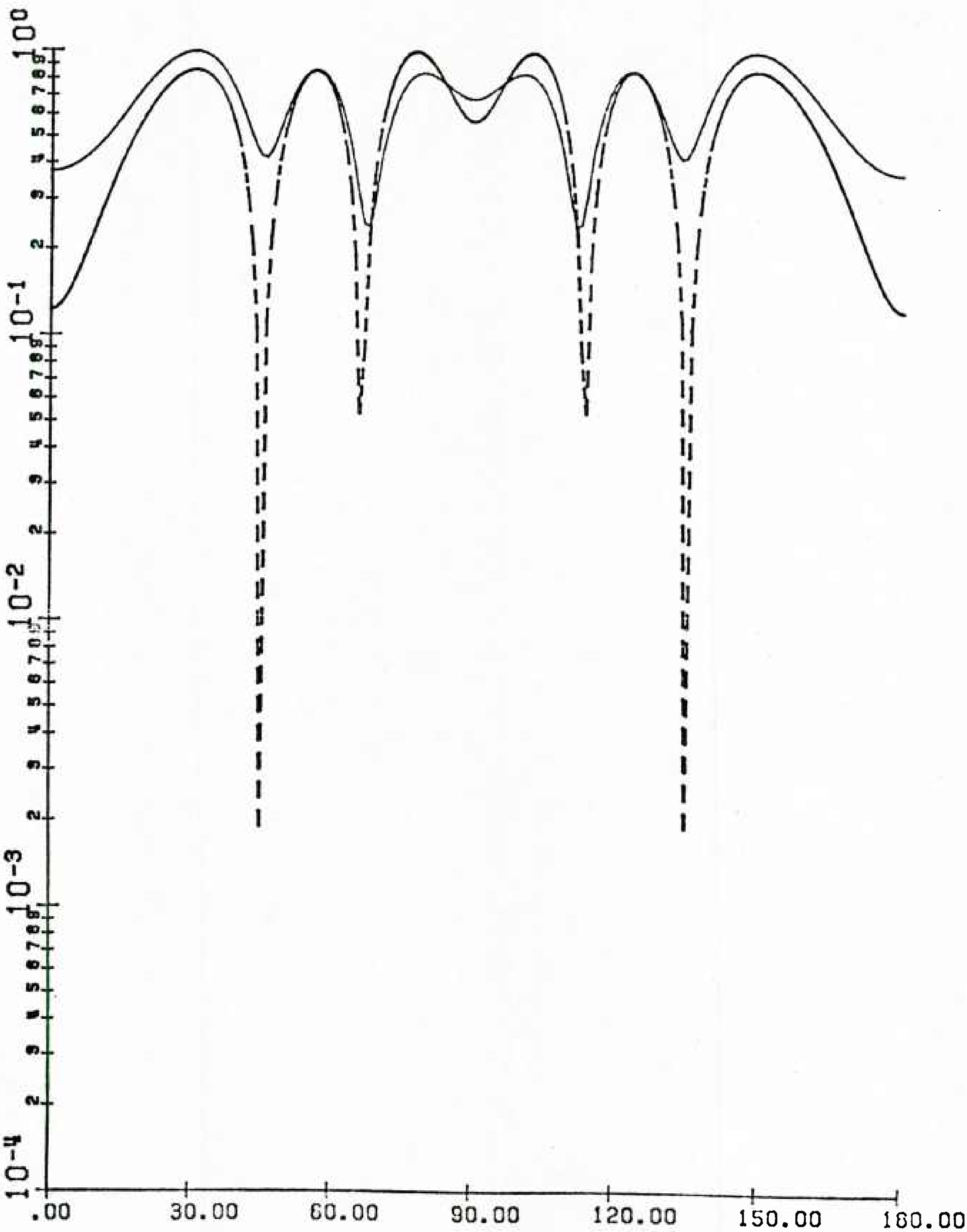


Fig. 17 Array factor in the xz plane of the seventy six antenna planar array with excitation to give a 135 degrees scan in the yz plane

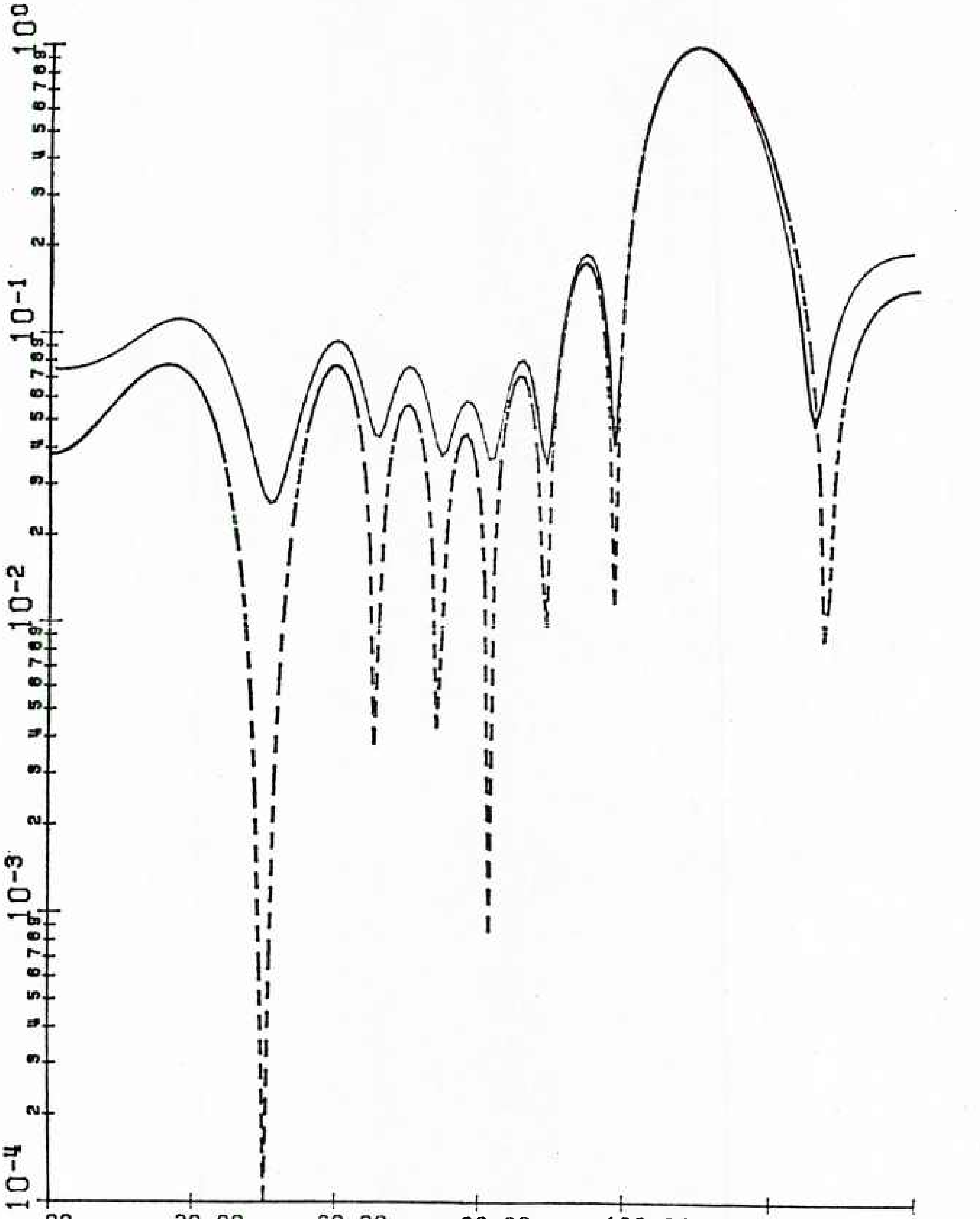


Fig. 18 Array factor in the yz plane of the seventy six antenna planar array with excitation to give a 135 degrees scan in the yz plane

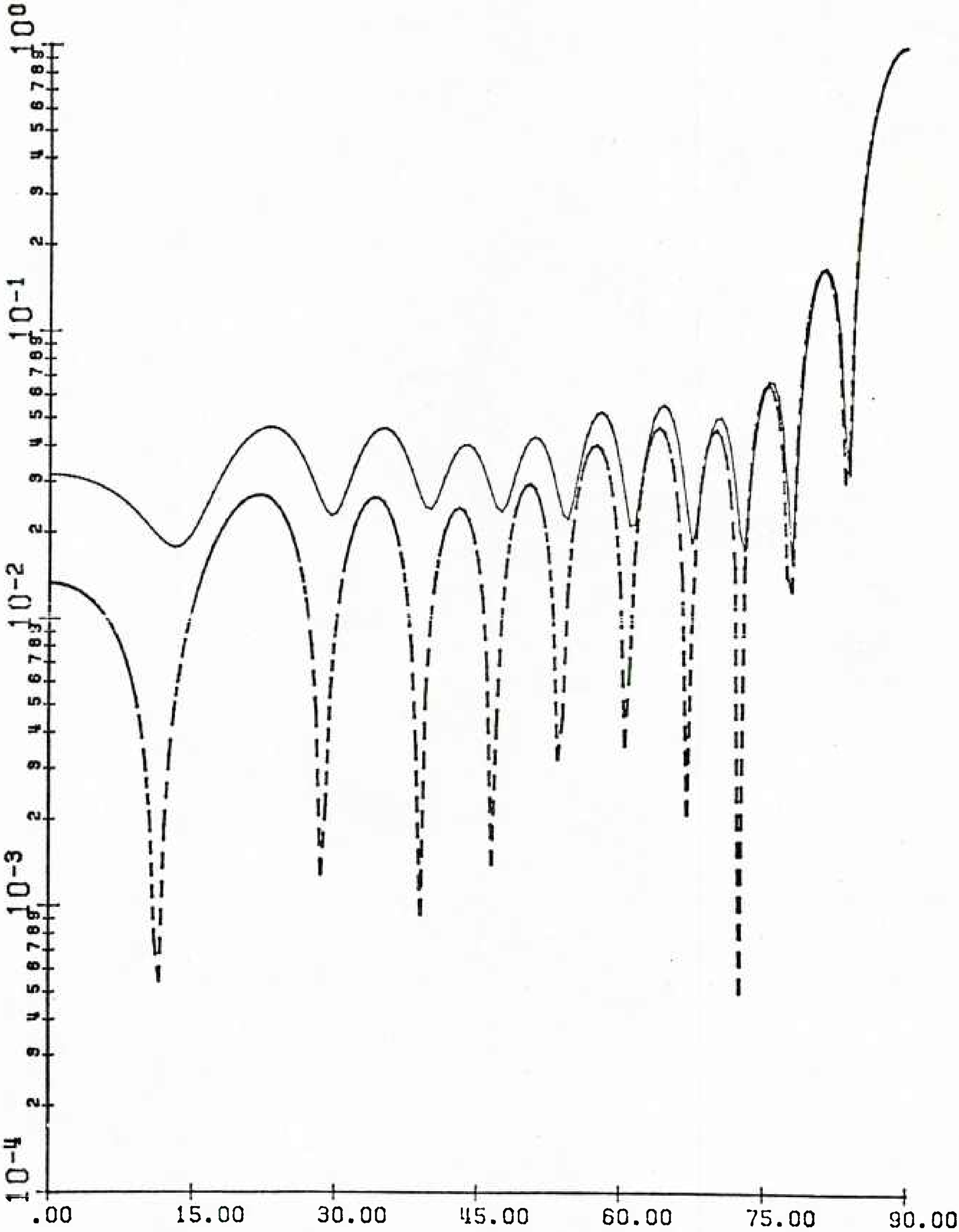


Fig. 19 Array factor in the xz plane of the three hundred and seventy two antenna planar array with uniform excitation

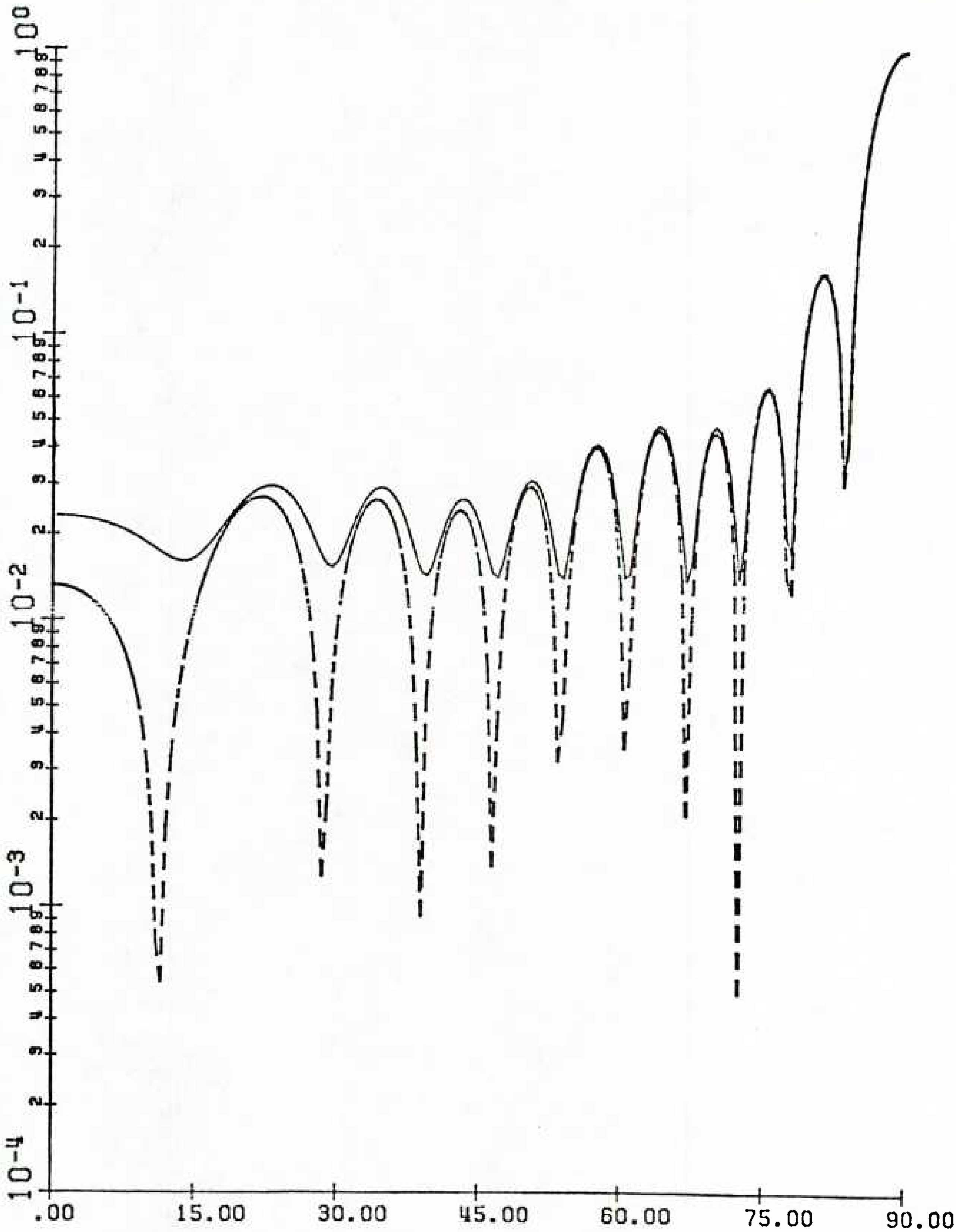


Fig. 20 Array factor in the yz plane of the three hundred and seventy two antenna planar array with uniform excitation

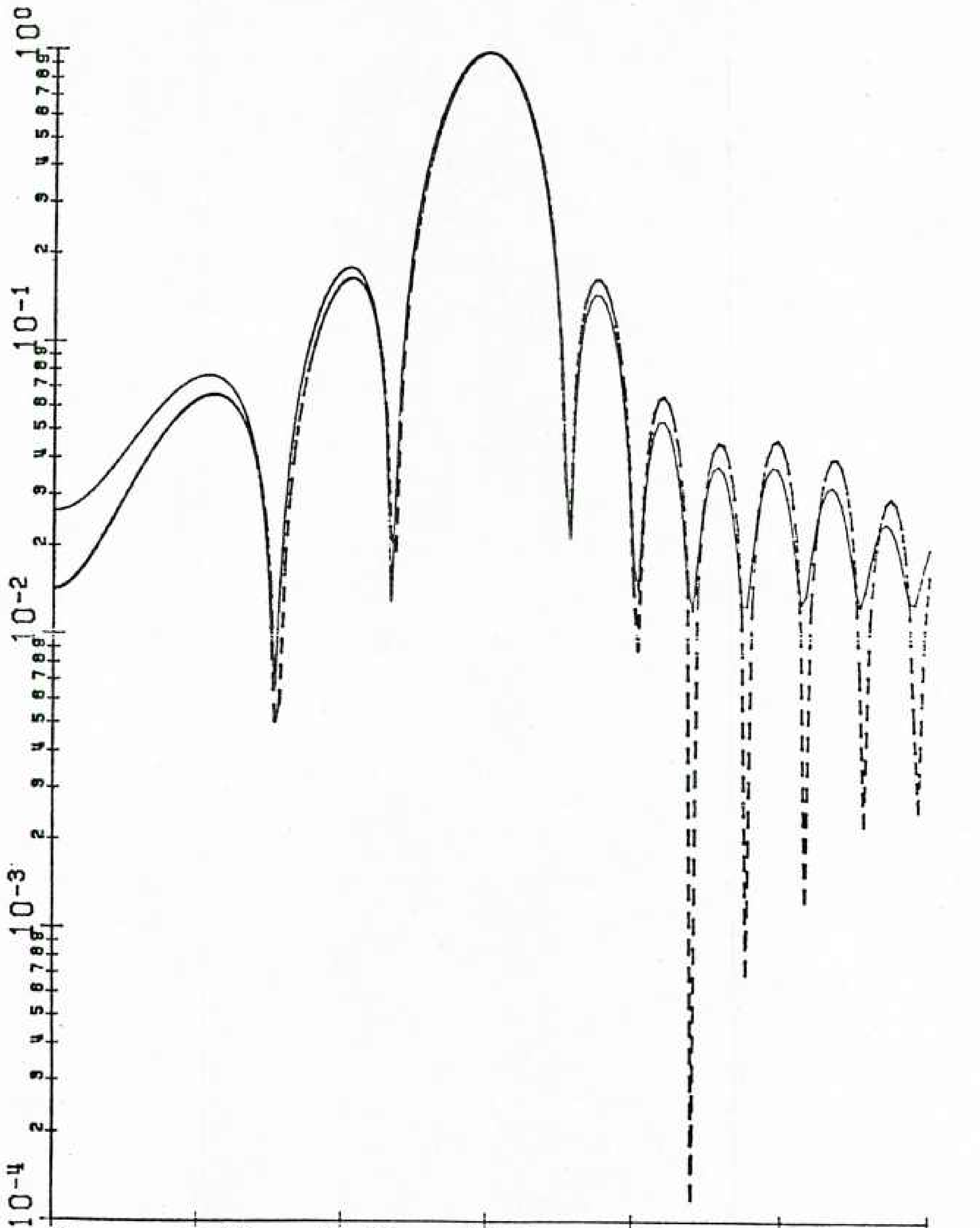


Fig. 21 Array factor in the xz plane of the three hundred and seventy two antenna planar array with excitation to give 45 degrees scan in the xz plane

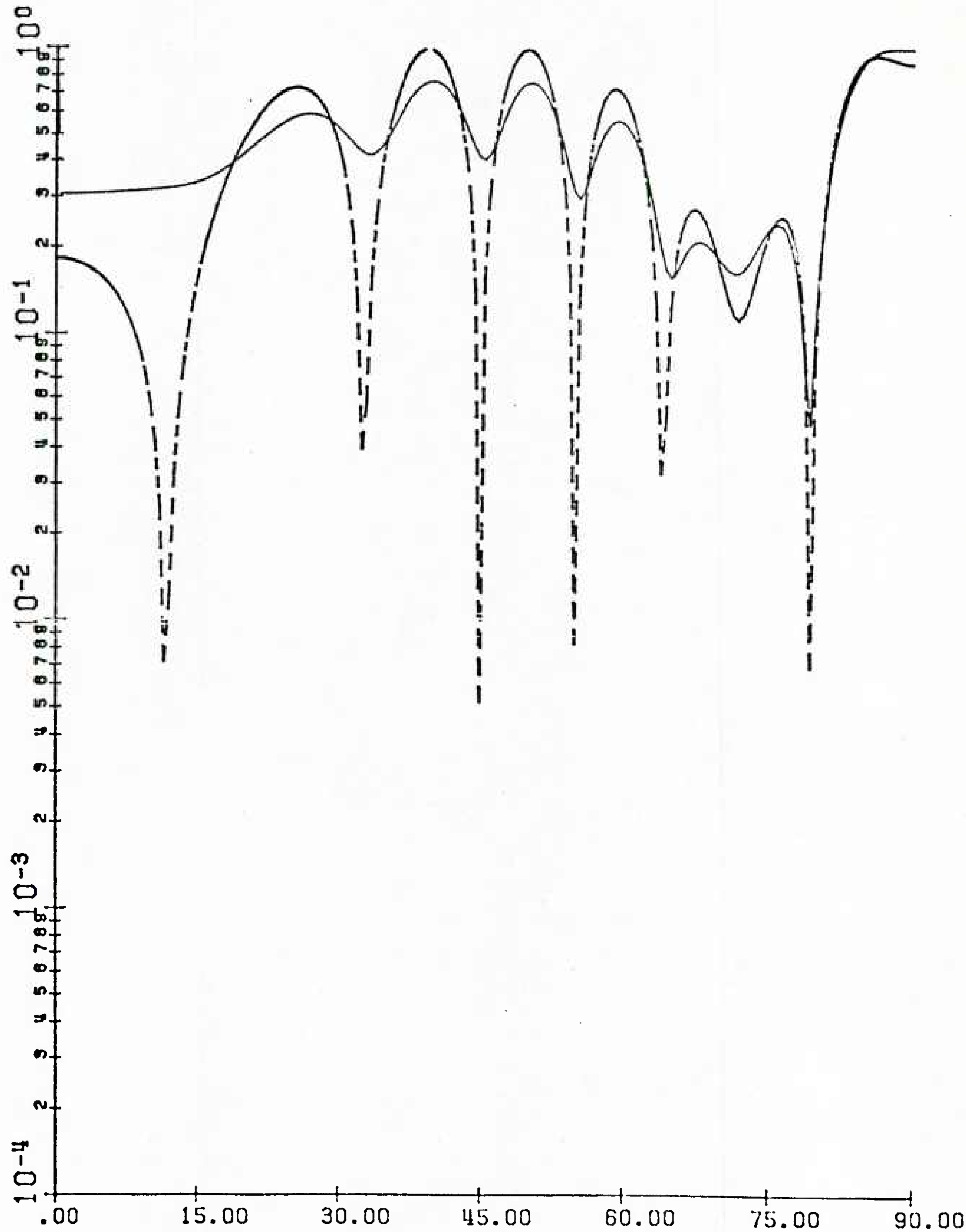


Fig. 22 Array factor in the yz plane of the three hundred and seventy two antenna planar array with excitation to give 45 degrees scan in the xz plane

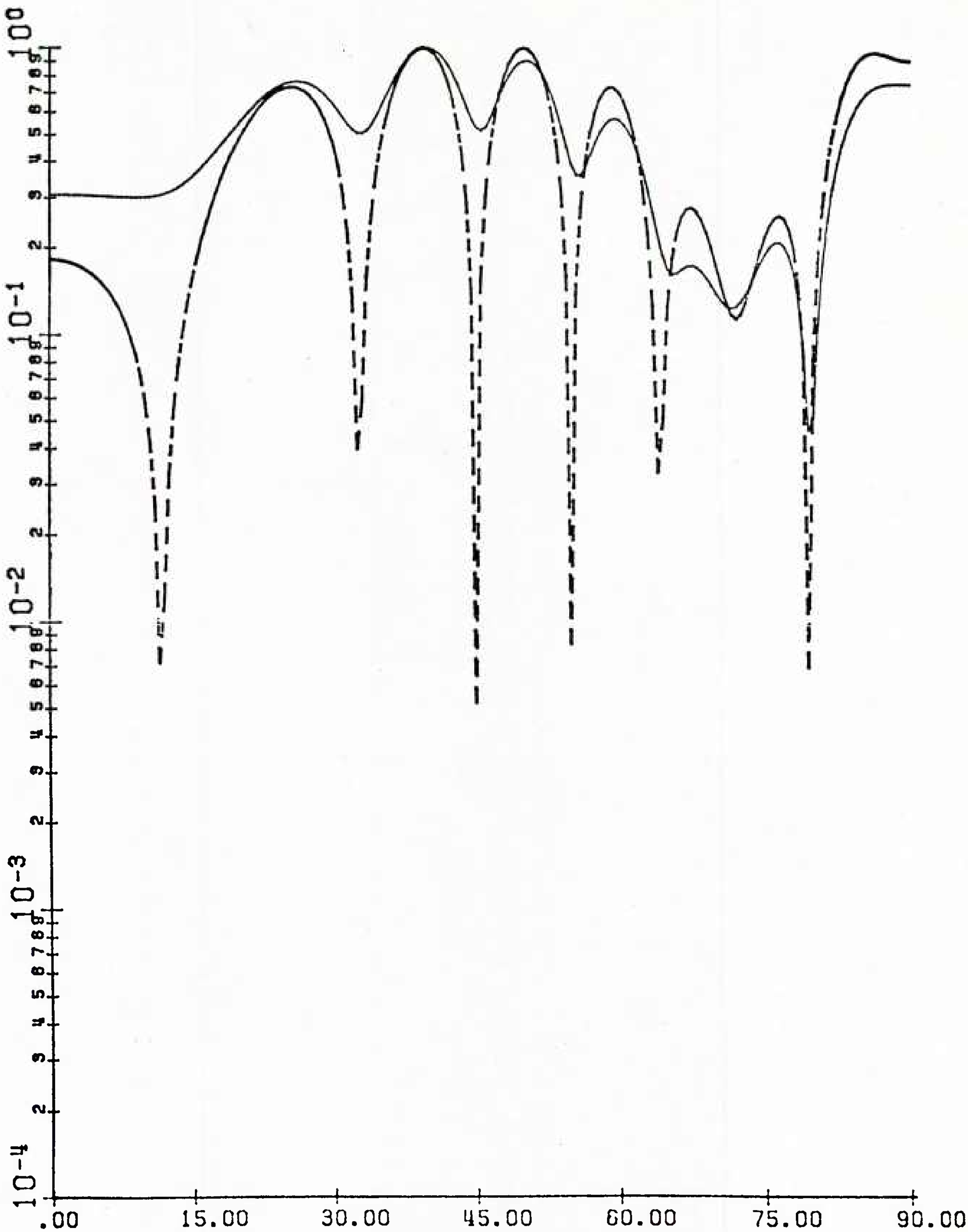


Fig. 23 Array factor in the xz plane of the three hundred and seventy two antenna planar array with excitation to give 135 degrees scan in the yz plane

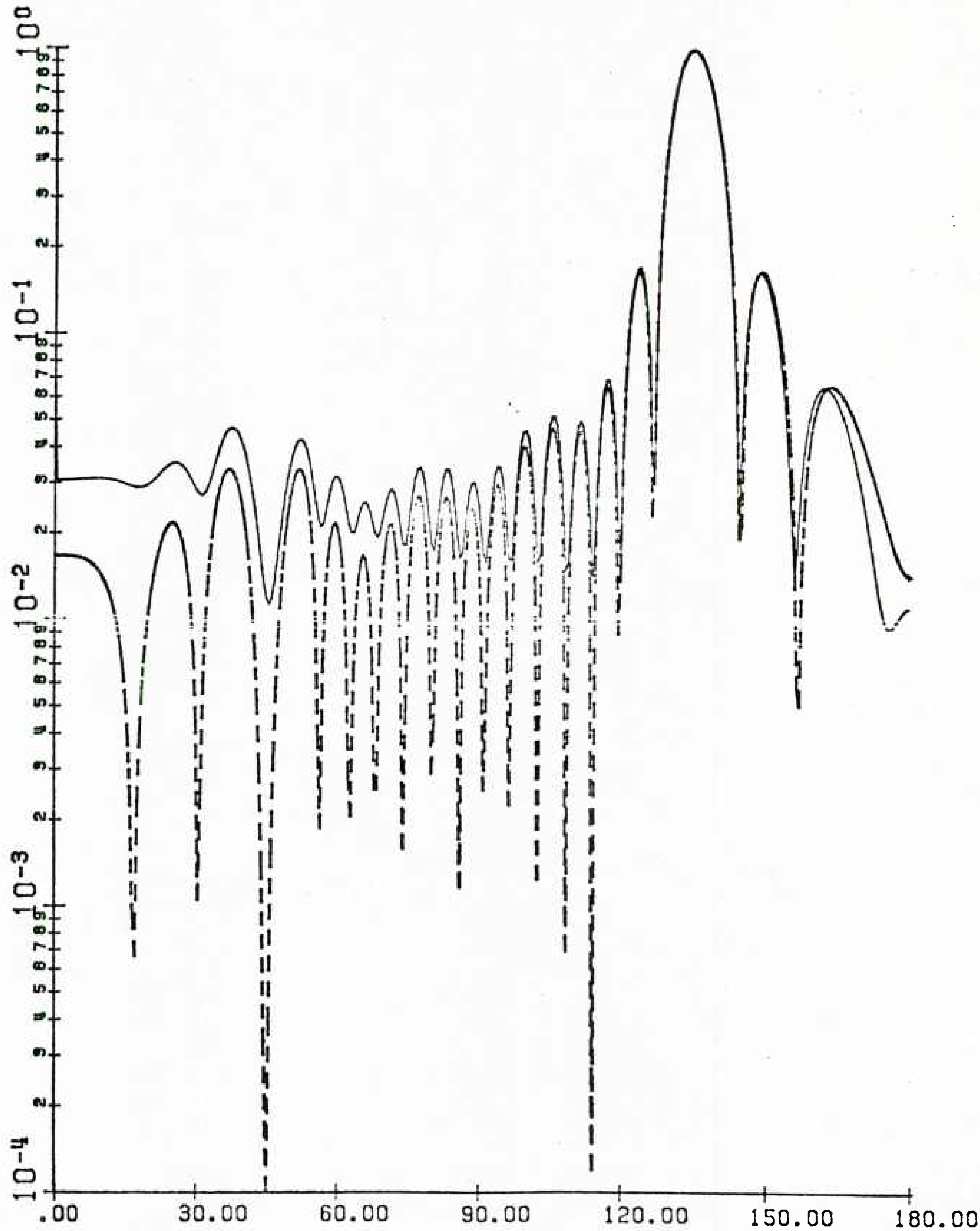


Fig. 24 Array factor in the yz plane of the three hundred and seventy two antenna planar array with excitation to give 135 degrees scan in the yz plane

field pattern of an element with only one expansionn function. This is the array factor used in the graphs discussed above.

$$(NAF)_n = \frac{(FP)_n}{(EP)_1} \qquad (2)$$

Here $(NAF)_n$ is the normalized array factor for the DCM solution using n expansions per antenna element. $(FP)_n$ is the far field pattern for the DCM solution using n expansion functions per antenna element. $(EP)_1$ is the element pattern or the far field pattern of an element with only one expansion function per element. Since both $(FP)_n$ and $(EP)_1$ goes to zero when the angles are either 0 or 180 degrees, $(NAF)_n$ is indeterminate at those angles.

As we can see from the graphs, for larger arrays, the main beam is not effected by the mutual coupling but the side lobes and the nulls are effected strongly. In addition we can see that the difference between the actual and the ideal array factors are more pronounced for the 45 degrees beam steering in the xz plane. This is expected since the coupling between the antenna elements that are adjacent in the x-direction is stronger than the coupling between the antenna elements that are adjacent in the y-direction. Also, we can see that the array factors computed from the solutions using one expansion function per antenna and the

array factors computed from the solutions using three expansion functions per antenna are surprisingly close. This indicates that at least for the given spacings and antenna sizes, the solutions using only one expansion function per antenna may be good enough for most purposes.

III. CONCLUSIONS

Three more types of problems suggested in the first report on DCM [1] are solved and the number of iterations needed are found to be reasonably small for the antenna array problems. The helix problems required more iterations but the number of iterations needed for the required degree of accuracy is found to be practically independent of the length of the helix.

Three other major types of problems formulated or suggested in [1] still remains to be solved. They are

(i) scattering from a arbitrarily shaped planer conducting surfaces or dielectric sheets

(ii) scattering from a imperfectly conducting or dielectric bodies

(iii) planar array antenna backed by a finite sized ground plane

The major question here is wherether or not convergence will be achieved for these types of problems. The condition for convergence is that the absolute value of the largest

eigenvalue of $[\mathcal{Z}]$ must be less then 1 as shown in [1]. There are some practical applications where the solutions of these types of problems are useful. Therefore, solving some representative problems are suggested as possible future extension to this report.

APPENDIX A

SPATIAL DOMAIN INTERPRETATION OF THE SPECTRAL ITERATION TECHNIQUES

Both the spectral solution techniques(STD,SIT etc.) [2], [3], [4], [5] and the Discrete Convolution Method (DCM) [1] use the discrete Fourier Transform to solve electromagnetic fields problems. The following spatial domain interpration of the spectral solution techniques may give a better understanding of the differences between the methods.

The electric field integral equation for a perfectly conducting scatterer or radiator is [2]

$$(\overline{\overline{G}} * \vec{J})_t = -\vec{E}_t^i \qquad r \in S \qquad \text{(A1)}$$

Here $\overline{\overline{G}}$ is the Green's Dyadic, $\vec{J}$ is the surface current density on surface S and $\vec{E}_t^i$ is the tangential component of the impressed field. We can extend the field equation over all space so that (A1) becomes

$$\overline{\overline{G}} * \vec{J} = -\vec{E}^i + \vec{F} \qquad \text{(A2)}$$

where $\vec{F}$ is the field outside the surface S. The Fourier tranformed version of (A2) reads

$$\tilde{\tilde{G}}(\Omega)\ \tilde{\tilde{J}}(\Omega) = -\tilde{\tilde{E}}_I(\Omega) + \tilde{\tilde{F}}(\Omega) \qquad \text{(A3)}$$

$$\Omega = (\omega^x, \omega^y, \omega^z) \text{ in general}$$

A formal solution to (A3) is

$$\tilde{\tilde{J}}(\Omega) = \tilde{\tilde{G}}^{-1}(\Omega)\ \tilde{\tilde{E}}(\Omega) \qquad \text{(A4)}$$

where $\tilde{\tilde{E}}(\Omega)$ is $-E_I(\Omega)+F(\Omega)$.

The spectral solution techniques essentially consists of iterative techniques that find $\vec{F}$ and thus $\vec{J}$. However since $\vec{F}$ cannot be found in closed form, the solution is a numerical solution. Therefore (A4) and hence (A3) is solved only for a finite number of specific discrete frequencies. The solution we get is an exact solution of (A3) and hence of (A1) only if $\vec{F}$ is known exactly and if the solution is for all frequencies. In the following discussion we will set aside the question of inaccuracies due to inexact knowledge of $\vec{F}$ because it is a question of convergence. Here we are discussing spatial domain interpretation rather than the convergence of the solution. Therefore

(a) since (A3) is satisfied only for discrete frequencies, we are sampling in frequency domain and we are solving the following set of equations

$$\tilde{\tilde{G}}(\Omega)\tilde{\tilde{J}}(\Omega)\,\delta\,(\Omega-\Omega_n) = \tilde{\tilde{E}}(\Omega)\,\delta\,(\Omega-\Omega_n) \qquad \text{(A5)}$$

$$\begin{aligned} \Omega_n &= (\omega_a^x, \omega_b^y, \omega_c^z) \\ \omega_a^x &= \omega_0^x, \omega_1^x, \ldots \\ \omega_b^y &= \omega_0^y, \omega_1^y, \ldots \\ \omega_c^z &= \omega_0^z, \omega_1^z, \ldots \end{aligned}$$

instead of (A3).

(b) Since we cannot solve for infinite number of discrete frequencies, we solve for only a finite number. This amounts to truncating in frequency domain. If we denote the truncating function by $\tilde{\tilde{H}}(\Omega)$, we are solving the following set of equations

$$\tilde{\tilde{H}}(\Omega)\tilde{\tilde{G}}(\Omega)\tilde{\tilde{J}}(\Omega)\,\delta(\Omega-\Omega_n) = \tilde{\tilde{H}}(\Omega)\tilde{\tilde{E}}(\Omega)\,\delta(\Omega-\Omega_n) \quad \text{(A6)}$$

instead of either (A3) or (A5).

Before we give the spatial domain interpretation of (A6), we will first examine the errors due to above approximations from a frequency domain perspective. It is well known [6] that sampling in the spatial domain will lead to aliasing in frequency domain if the function sampled is not limited in frequency. By reason of symmetry, we can easily show that sampling in frequency domain will lead to aliasing in the spatial domain if the function sampled is not limited in space. In solving (A3), we used sampling on both sides of the equation. The surface current $\vec{J}$ is indeed limited in space (being confined to conductor surface S) but

the field $\vec{E}$ is not. Therefore, there will be aliasing irrespective of the sampling rate. However, since the field $\vec{E}$ decays outside the surface, aliasing effect can be reduced by a high enough sampling rate so that the overlap occurs in the region where $\vec{E}$ has already decayed to a small value. Truncation of the frequency range by $\tilde{\tilde{H}}(\Omega)$ (the windowing function) on both sides of the equation leads to Gibb's phenomenon. This windowing error can be reduced by using different windowing strategies discussed in [6]. A good example is the Hamming windowing function.

We will now look at the above errors from a spatial domain perspective. In spectral solution techniques, solution to (A6) is found by using the Discrete Fourier Transform. This is possible if Ω_n's are chosen at equally spaced intervals. It can be proved [6] that (A6) in this case is equivalent to the Discrete Convolution equation

$$\bar{\bar{\mathcal{G}}}(K) * \vec{J}(K) = \vec{E}(K) \qquad K=(k_x,k_y,k_z) \tag{A7}$$

$$k_x = 0,1,\ldots,L$$

$$k_y = 0,1,\ldots,M$$

$$k_z = 0,1,\ldots,N$$

Up to this point, we have been discussing the general multi-dimensional case. From here on however, we will confine our discussion to the one dimensional case. Discussion for two or three dimensional problems would be similar. (We will use lower case indices instead of higher

case, to indicate one dimensionality.) Since $\vec{E}_I$ is known exactly and $\vec{F}$, once we achieve convergence, is also known exactly, $\vec{E}(k)$ represents the sampled value at k^{th} point of the exact field $\vec{E}$ [2], [5]. Also, $\vec{J}$ is the required solution and hence $\vec{J}(k)$ represents the sampled value at k^{th} point of the current $\vec{J}$. But $\bar{\bar{\mathcal{G}}}(k)$ is not the sampled value of $\bar{\bar{G}}$. Instead, $\bar{\bar{\mathcal{G}}}(k)$ is the inverse Discrete Fourier Transform (based on N+1 points) of $\tilde{\bar{\bar{G}}}(\omega)$.

$$\bar{\bar{\mathcal{G}}}(k) = \sum_{n=-N/2}^{N/2} \tilde{\bar{\bar{G}}}(n)\ e^{j\left(\frac{2\pi}{N+1}\right)kn} \qquad \text{(A8)}$$

Therefore $\bar{\bar{\mathcal{G}}}(k)$ is the inverse Fourier Transform of the sampled and truncated (or in general windowed) $\tilde{\bar{\bar{G}}}(\omega)$.

It is well known [6] that sampling in spatial domain will lead to a function in frequency domain which consists of periodic repetitions of the original frequency domain function. By reason of symmetry between the domains, we can easily show that sampling in frequency domain will lead to a function in spatial domain which consists of periodic repetitions of the original spatial domain function. Therefore, sampling of $\tilde{\bar{\bar{G}}}(\omega)$ will cause the inverse of the sampled function $\bar{\bar{\mathcal{G}}}_s$ to consist of periodic repetitions of the original spatial domain function $\bar{\bar{G}}$. This will cause overlaps as shown in Fig. 25 for the one dimensional case. But G is the Green's Dyadic, the field due to a unit impulse

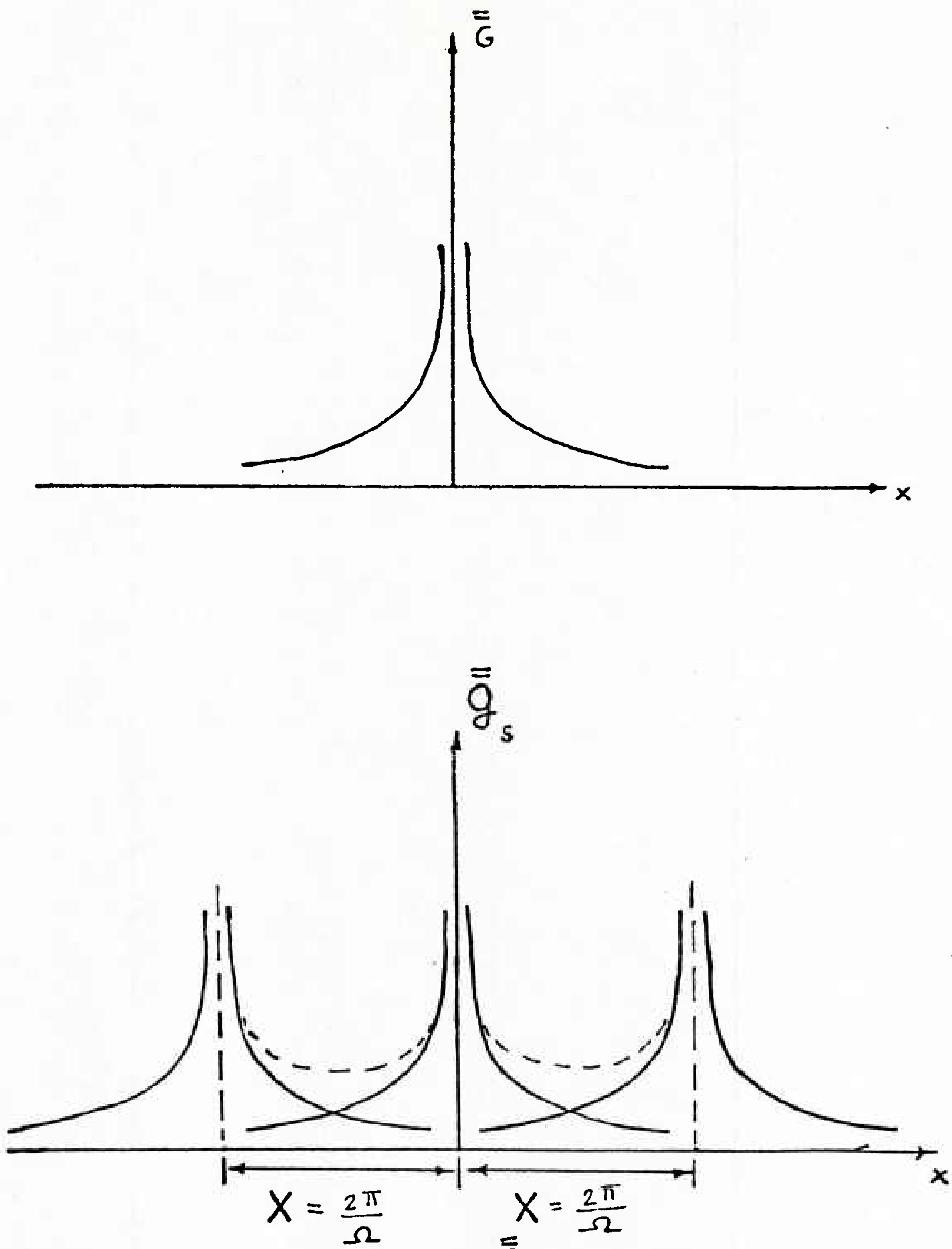


Fig. 25 Relationship between $\bar{\bar{g}}_s$ and $\bar{\bar{G}}$ for the one dimensional case

current at the origin. Therefore $\bar{\bar{g}}_s$ is the overlapped approximate field of a unit impulse current at the origin. Using higher sampling rates will cause the rate of repetition to decrease, i.e., increase X. Since we are interested in solving (A7) for only a finite region in space, error due to overlap will be insignificant if the sampling rate is high enough.

However, in addition to sampling, we also truncate the frequency function. This truncation will, in general, be done by using a windowing function $\tilde{\bar{\bar{H}}}(\omega)$. Therefore $\bar{\bar{G}}(k)$ can be thought of as the inverse Fourier Transform of the sampled version of the function $\tilde{\bar{\bar{G}}}(\omega)\tilde{\bar{\bar{H}}}(\omega)$. $\tilde{\bar{\bar{G}}}(\omega)$ is the Fourier Transform of the Green's Dyadic $\bar{\bar{G}}$, the field due to a unit current impulse at the origin. Therefore $\tilde{\bar{\bar{G}}}(\omega)\tilde{\bar{\bar{H}}}(\omega)$ is the Fourier Transform of the field due to a current distribution $\vec{H}$, where $\vec{H}$ is the inverse Fourier Transform of the windowing function $\tilde{\bar{\bar{H}}}(\omega)$. Therefore $\bar{\bar{g}}(k)$ is the sampled value at k^{th} point of the field due to the current distribution $\vec{H}$. Therefore, the result of the convolution product $\bar{\bar{g}}*\bar{J}$ is the (overlapped) approximate field due to the sampled values of the current $\vec{J}$ multiplied by the current distribution $\vec{H}$. Since a convolution product produces a finite set of values, each one corresponding to a point in space, each value can be interpreted as the approximate field at the corresponding point in space due to the current $\vec{J}$, expanded in terms of the basis or expansion function $\vec{H}$. Since solving (6) amounts to matching the left

hand side and the right hand side of (A7) [6], the solution of (A6) can now be seen as point matching the field E with the approximate field due to $\vec{J}$ expanded in terms of the basis or expansion function $\vec{H}$. If the error due to overlap is not significant, then the solution of (A6) is equivalent to point matching the known field E with the field due to $\vec{J}$ expanded in terms of the basis or expansion function $\vec{H}$. Therefore, the spectral iteration method is equivalent to the Method of Moments [7], if we use the inverse Fourier Transform of the windowing function $\tilde{\bar{H}}(\omega)$ as expansion function and point match. The only difference is that with the Method of Moments, Gaussian elimination is normally used instead of iterative techniques.

The Discrete Convolution Method [1] on the other hand, is the iterative solution of the matrix equation formulated by the Method of Moments. Therefore the difference between the Discrete Convolution Method and the spectral iteration techniques is that with the former method, the expansion function is chosen explicitly whereas with the latter, the expansion function is chosen implicitly; i.e. the inverse Fourier Transform of the windowing function.

APPENDIX B

COMPUTER PROGRAMS

The computer programs given in this section are written to solve the three additional types of problems. Although they are not optimized in any sense of the word, they are written to avoid obvious wastes of computing time and memory space and is fairly efficient.

I. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE HELIX PROBLEM

Both the main program segment and the subroutine CALZ are from [8]. The slight modifications in the main program are self-explaining so no attempt will be made here to give a description. The subroutine HELIX on the other hand, is written to compute the co-ordinates of the helix points from the given radius, pitch, number of turns, and the number of points for which the co-ordinates must be found.

```
C
C       THIS IS THE MAIN PROGRAM
C
        COMPLEX Z(3600),U(400),C(400),E(3),EI(2),UV(4)
        COMPLEX U1,ZL(30),ZIN,YIN,V,ZC
        COMPLEX TOE(1024),EX(400),CUR(1024),GUESS
        COMMON /G/GUESS,CUR
        COMMON XX(800),XY(800),XZ(800),TX(800),TY(800),TZ(800),AL(800)
        COMMON T(1600),TP(1600),BK,RAD2(21),L(11),LR(21)/COA/C
        DIMENSION PX(900),PY(900),PZ(900),LL(11),RAD(21),IFP(60)
        DIMENSION LP(30)
        PI=3.141592654
        OPEN(UNIT=21,FILE='ARDATA.DAT')
 1      READ(1,101) NW,NP,NR,BK,IFLAG
        WRITE(3,102) NW,NP,NR,BK
        LL(NW+1)=801
        LR(NR+1)=801
        IF(IFLAG.NE.0) GO TO 300
        READ(1,130)  (PX(I),I=1,NP)
        READ(1,130)  (PY(I),I=1,NP)
        READ(1,130)  (PZ(I),I=1,NP)
        GO TO 301
  300   CONTINUE
        IF(IFLAG.NE.1) GO TO 310
        CALL HELIX(PX,PY,PZ,NP)
        GO TO 301
  310   CONTINUE
        READ(1,130)WIREL,SEGL
        DO 501 I=1,NP
        PX(I)=WIREL
        WIREL=WIREL+SEGL
        PY(I)=0.0
        PZ(I)=0.0
 501    CONTINUE
  301   CONTINUE
        WRITE(3,104) (PX(I),I=1,NP)
        WRITE(3,105) (PY(I),I=1,NP)
        WRITE(3,106) (PZ(I),I=1,NP)
        READ(1,107)  (LL(I),I=1,NW)
        WRITE(3,108) (LL(I),I=1,NW)
        READ(1,107)  (LR(I),I=1,NR)
        WRITE(3,103) (LR(I),I=1,NR)
        READ(1,109)  (RAD(I),I=1,NR)
        WRITE(3,110) (RAD(I),I=1,NR)
 101    FORMAT(3I3,E14.7)
 102    FORMAT('ONW NP NR     BK'/3I3,E14.7)
 130    FORMAT(10F0.0)
 104    FORMAT('0PX'/(1X,8F8.4))
 105    FORMAT('0PY'/(1X,8F8.4))
 106    FORMAT('0PZ'/(1X,8F8.4))
 107    FORMAT(20I3)
 108    FORMAT('0LL'/(1X,10I4))
 109    FORMAT(5E14.7)
 103    FORMAT('0LR'/(1X,10I4))
 110    FORMAT('0RAD'/(1X,8E0.0))
        DO 46 I=1,NR
        RAD2(I)=RAD(I)*RAD(I)
 46     CONTINUE
        J1=1
        J2=2
```

```
      N1=0
      DO 2 J=1,NP
      IF(LL(J1)-J) 3,4,3
    4 J2=J2-1
      L(J1)=J2
      J1=J1+1
      GO TO 2
    3 N1=N1+1
      J3=J-1
      IF((N1/2*2-N1).EQ.0) J2=J2+1
      XX(N1)=.5*(PX(J)+PX(J3))
      XY(N1)=.5*(PY(J)+PY(J3))
      XZ(N1)=.5*(PZ(J)+PZ(J3))
      S1=PX(J)-PX(J3)
      S2=PY(J)-PY(J3)
      S3=PZ(J)-PZ(J3)
      S4=SQRT(S1*S1+S2*S2+S3*S3)
      TX(N1)=S1/S4
      TY(N1)=S2/S4
      TZ(N1)=S3/S4
      AL(N1)=S4
    2 CONTINUE
      L(J1)=J2
      N=J2-2
      CALL CALZ(N,N1,Z)
      WRITE(21,113) (Z(I),I=1,N)
  113 FORMAT(5E14.7)
      DO 45 I=1,N
      U(I)=0.
   45 CONTINUE
      READ(1,107) NSET
      WRITE(3,127) NSET
      IF(NSET) 33,33,32
  127 FORMAT('ONSET'/I4)
   32 DO 26 III=1,NSET
      U1=(0.,1.)
      READ(1,126) THE,PHI,EI(1),EI(2)
  126 FORMAT(8E0.0)
      GUESS=GUESS*RAD(1)/188.365
      WRITE(3,125) THE,PHI,EI(1),EI(2)
  125 FORMAT('0**********'/'0THETA   PHI              EI(1)',
     1       '                    EI(2)'/2F6.0,4E11.4)
      A1=CABS(EI(1))**2+CABS(EI(2))**2
      A2=BK*BK
      TH=THE*.0174533
      PH=PHI*.0174533
      CT=COS(TH)
      ST=SIN(TH)
      CP=COS(PH)
      SP=SIN(PH)
      S1=CT*CP
      S2=CT*SP
      BK1=BK*ST*CP
      BK2=BK*ST*SP
      BK3=BK*CT
      J1=1
      J2=-2
      DO 27 J=1,N
      IF(L(J1)-J) 29,28,29
   28 J2=J2+2
```

```
      J1=J1+1
      KK=1
      GO TO 30
   29 UV(1)=UV(3)
      UV(2)=UV(4)
      KK=3
   30 DO 31 M=KK,4
      J3=J2+M
      XDT=TX(J3)*S1+TY(J3)*S2-TZ(J3)*ST
      XDP=-TX(J3)*SP+TY(J3)*CP
      BKR=XX(J3)*BK1+XY(J3)*BK2+XZ(J3)*BK3
      UV(M)=(XDT*EI(1)+XDP*EI(2))*(COS(BKR)+U1*SIN(BKR))
   31 CONTINUE
      J3=(J-1)*4
      J4=J3+1
      J5=J4+1
      J6=J5+1
      J7=J6+1
      U(J)=T(J4)*UV(1)+T(J5)*UV(2)+T(J6)*UV(3)+T(J7)*UV(4)
      J2=J2+2
   27 CONTINUE
      WRITE(21,113) (U(I),I=1,N)
   26 CONTINUE
   33 CONTINUE
      WRITE(3,112)
  112 FORMAT(1H ,'TYPE 1 TO STOP, ELSE 0 AND RETURN')
      READ(1,101) IFLAG
      IF(IFLAG.EQ.0) GO TO 1
      CLOSE(UNIT=21)
      STOP
      END
C
C     THIS IS SUBROUTINE #1
C
      SUBROUTINE CALZ(N,N1,Z)
      COMPLEX Z(3600),PSI(3200),U,U1,U2,U3,U4,U5,U6
      COMMON XX(800),XY(800),XZ(800),TX(800),TY(800),TZ(800),AL(800)
      COMMON T(1600),TP(1600),BK,RAD2(21),L(11),LR(21)
      DIMENSION DC(3200)
      U=(0.,1.)
      PI=3.141593
      ETA=376.7307
      C1=.125/PI
      C2=.25/PI
      J1=1
      J2=-2
      DO 1 J=1,N
      IF(L(J1)-J) 3,4,3
    4 J2=J2+2
      J1=J1+1
    3 J3=(J-1)*4
      J4=J3+1
      J5=J4+1
      J6=J5+1
      J7=J6+1
      K4=J2+1
      K5=K4+1
      K6=K5+1
      K7=K6+1
      S1=AL(K4)+AL(K5)
```

```
      S2=AL(K6)+AL(K7)
      T(J4)=AL(K4)*.5*AL(K4)/S1
      T(J5)=AL(K5)*(AL(K4)+.5*AL(K5))/S1
      T(J6)=AL(K6)*(AL(K7)+.5*AL(K6))/S2
      T(J7)=AL(K7)*.5*AL(K7)/S2
      TP(J4)=AL(K4)/S1
      TP(J5)=AL(K5)/S1
      TP(J6)=-AL(K6)/S2
      TP(J7)=-AL(K7)/S2
      J2=J2+2
 1    CONTINUE
      U3=U*BK*ETA
      U4=-U/BK*ETA
      BK2=BK*BK/2.
      BK3=BK2*BK/3.
      N9=0
      N2=1
      N0=1
      N3=-2
      DO 10 NS=1,N
      IF(L(N2)-NS) 12,11,12
 11   KK=1
      N3=N3+2
      N2=N2+1
      GO TO 13
 12   KK=3
      DO 14 NF=1,N1
      N4=NF+N1
      N5=N4+N1
      N6=N5+N1
      DC(NF)=DC(N5)
      DC(N4)=DC(N6)
      PSI(NF)=PSI(N5)
      PSI(N4)=PSI(N6)
 14   CONTINUE
 13   CONTINUE
      DO 15 K=KK,4
      N7=N3+K
      K1=(K-1)*N1
      N0=1
      DO 16 NF=1,N1
      IF(NF-LR(N0)) 5,6,5
 6    AA=RAD2(N0)
      N0=N0+1
 5    CONTINUE
      N8=NF+K1
      S1=XX(N7)-XX(NF)
      S2=XY(N7)-XY(NF)
      S3=XZ(N7)-XZ(NF)
      R2=S1*S1+S2*S2+S3*S3+AA
      R=SQRT(R2)
      RT=ABS(S1*TX(N7)+S2*TY(N7)+S3*TZ(N7))
      RT2=RT*RT
      RH=(R2-RT2)
      ALP=.5*AL(N7)
      AR=ALP/R
      S1=BK*R
      U2=COS(S1)-U*SIN(S1)
      IF(AR-.1) 22,22,21
 21   U2=U2*C1/ALP
```

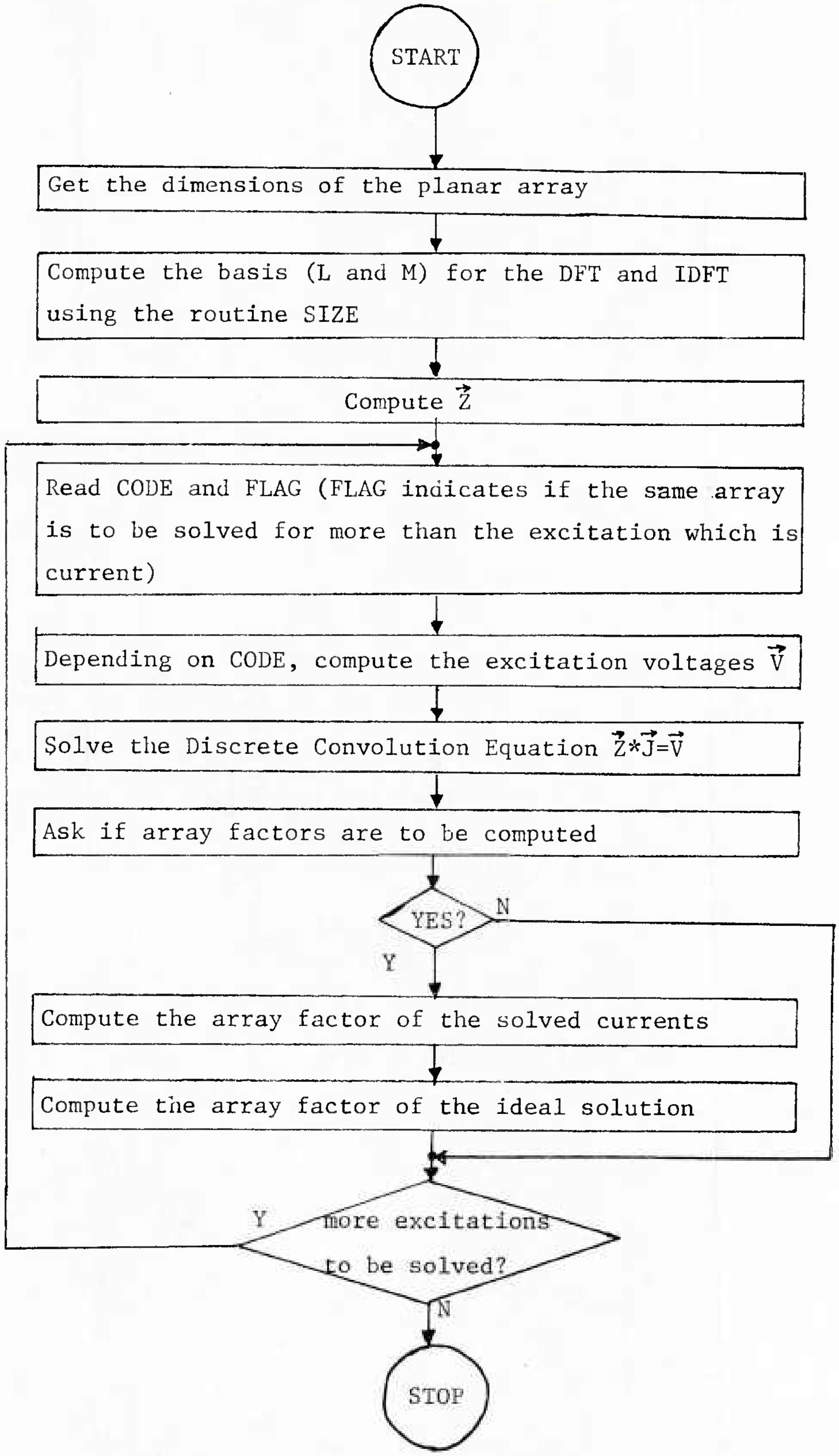
```
      S1=RT-ALP
      S2=RT+ALP
      S3=SQRT(S1*S1+RH)
      S4=SQRT(S2*S2+RH)
      IF(S1) 18,18,19
   18 AI1=ALOG((S2+S4)*(-S1+S3)/RH)
      GO TO 20
   19 AI1=ALOG((S2+S4)/(S1+S3))
   20 AI2=AL(N7)
      AI3=(S2*S4-S1*S3+RH*AI1)/2.
      AI4=AI2*(RH+ALP*ALP/3.+RT2)
      S3=AI1*R
      S1=AI1-BK2*(AI3-R*(2.*AI2-S3))
      S2=-BK*(AI2-S3)+BK3*(AI4-3.*AI3*R+R2*(3.*AI2-S3))
      GO TO 28
   22 U2=U2*C2/R
      BA=BK*ALP
      BA2=BA*BA
      AR2=AR*AR
      AR3=AR2*AR
      ZR=RT/R
      ZR2=ZR*ZR
      ZR3=ZR2*ZR
      ZR4=ZR3*ZR
      H1=(3.-30.*ZR2+35.*ZR4)*AR3/40.
      A1=AR*(-1.+3.*ZR2)/6.+(3.-30.*ZR2+35.*ZR4)*AR3/40.
      A0=1.+AR*A1
      A2=-ZR2/6.-AR2*(1.-12.*ZR2+15.*ZR4)/40.
      A3=AR*(3.*ZR2-5.*ZR4)/60.
      A4=ZR4/120.
      S1=A0+BA2*(A2+BA2*A4)
      S2=BA*(A1+BA2*A3)
   28 PSI(N8)=U2*(S1+U*S2)
      DC(N8)=TX(NF)*TX(N7)+TY(NF)*TY(N7)+TZ(NF)*TZ(N7)
   16 CONTINUE
   15 CONTINUE
      N3=N3+2
      J3=(NS-1)*4
      J7=-2
      J9=1
      DO 25 NF=1,N
      J1=(NF-1)*4
      IF(L(J9)-NF) 26,27,26
   27 J9=J9+1
      J7=J7+2
   26 N9=N9+1
      U5=0.
      U6=0.
      J5=0
      DO 23 JS=1,4
      J4=J3+JS
      J8=J5+J7
      DO 24 JF=1,4
      J6=J8+JF
      J2=J1+JF
      U5=T(J2)*T(J4)*DC(J6)*PSI(J6)+U5
      U6=TP(J2)*TP(J4)*PSI(J6)+U6
   24 CONTINUE
      J5=J5+N1
   23 CONTINUE
```

```
          Z(N9)=U5*U3+U6*U4
          J7=J7+2
          IF(N9.GE.N)RETURN
 25       CONTINUE
 10       CONTINUE
          RETURN
          END
          SUBROUTINE HELIX(X,Y,Z,NP)
          REAL X(1),Y(1),Z(1),RAD,PITCH,PHI,DPHI
          INTEGER TURNS
          READ(1,100) RAD,PITCH,TURNS
          WRITE(3,110)RAD,PITCH,TURNS
  100     FORMAT(2F0.0,I0)
  110     FORMAT(1H ,'RADIUS=',E14.7/
     $          1H ,'PITCH =',E14.7/
     $          1H ,'NO OF TURNS=',I7)
          PITCH=PITCH/6.2831853
          PHI=0.0
          DPHI=6.2831853*TURNS/FLOAT(NP-1)
          DO 10 I=1,NP
          X(I)=RAD*COS(PHI)
          Y(I)=RAD*SIN(PHI)
          Z(I)=PITCH*PHI
          PHI=PHI+DPHI
   10     CONTINUE
          RETURN
          END
```

II. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEMS (SINGLE EXPANSION FUNCTION PER ANTENNA ELEMENT)

Subroutine SHAPE zerorize the corners i.e. it zerorize the phantom antenna element currents. Subroutine PATTRN computes the array factor for the given set of array currents. The main program segment solves the planar array problems using the solution routine SOLVE and other routines not listed in this section. They are listed in section III since these routines are common to both single and multiple expansion solution programs.

The flow chart for the main program segment is as follows.



START
Get the dimensions of the planar array
Compute the basis (L and M) for the DFT and IDFT using the routine SIZE
Compute $\vec{Z}$
Read CODE and FLAG (FLAG indicates if the same array is to be solved for more than the excitation which is current)
Depending on CODE, compute the excitation voltages $\vec{V}$
Solve the Discrete Convolution Equation $\vec{Z}*\vec{J}=\vec{V}$
Ask if array factors are to be computed
YES?
N
Y
Compute the array factor of the solved currents
Compute the array factor of the ideal solution
Y
more excitations to be solved?
N
STOP

```
C     PROGRAM TO COMPUTE MUTUAL, IMAGE AND TOTAL IMPEDANCES
      LOGICAL EXCITE
      INTEGER CODE,FLAG,POSPTR,FLAG2
      COMPLEX MUTUAL,IMAGE,ZMNG,V(1849),Z(16384),CZERO,Y(1849)
C     READ IN NROWS - NUMBER OF ROWS OF ANTENNA ELEMENTS
C             NCOLS - NUMBER OF COLUMNS
C              DX   - SEPERATION BETWEEN ELEMENTS IN X DIRECTION
C              DY   - DISTANCE OF ELEMENTS FROM GROUND PLANE
C              DZ   - SEPERATION BETWEEN ELEMENTS IN Z DIRECTION
      EXCITE=.TRUE.
      TWOPI=6.2831853
      CZERO=(0.0,0.0)
      OPEN(UNIT=21,FILE='PATTRN.DAT')
      READ(1,100) NSIDE,CODE,DX,DY,DZ,WLNGTH,RAD
      NT=NSIDE*3-2
      NROWS=NT
      NCOLS=NT
      CALL SIZE(NCOLS,NROWS,LM,MM,L,M,N)
      DO 1 I=1,N
      Z(I)=CZERO
    1 CONTINUE
      DX=DX*TWOPI
      DY=DY*TWOPI
      DZ=DZ*TWOPI
      WLNGTH=WLNGTH*TWOPI
      RAD=RAD*TWOPI
      NELEM=NROWS*NCOLS
      ZELEM=0.0
      DY2SQ=4.0*DY*DY
      RAD2=RAD*RAD
      WLBY2=WLNGTH/2.0
      POSPTR=1
      DO 20 I=1,2*NT-1
      XCUR=(I-1)*DX
      ZCUR=(6*NSIDE-5-I)*DZ
      DO 10 J=1,2*NT-1
      RR=XCUR*XCUR+DY2SQ
      IMAGE=ZMNG(ZELEM,ZCUR,WLBY2,WLBY2,RR)
      RR=RR-DY2SQ
      IF(RR.EQ.0.0) RR=RR+RAD2
      MUTUAL=ZMNG(ZELEM,ZCUR,WLBY2,WLBY2,RR)
      Z(POSPTR)=MUTUAL-IMAGE
      POSPTR=POSPTR+1
      XCUR=XCUR-DX
      ZCUR=ZCUR-DZ
   10 CONTINUE
      POSPTR=POSPTR+L-NT-NT+1
   20 CONTINUE
   30 READ(1,100) CODE,FLAG
      IF(CODE.EQ.1) CALL VOLTU(NCOLS,NROWS,V)
      IF(CODE.EQ.2) CALL VOLTC(NCOLS,NROWS,V)
      IF(CODE.EQ.3) CALL TAP(NCOLS,NROWS,DX,DZ,V)
      IF(CODE.EQ.4) CALL VOLTK(NCOLS,NROWS,V)
      IF(CODE.EQ.5) CALL STEER(NCOLS,NROWS,DX,DZ,V)
      IF(CODE.EQ.6) CALL PHASE(NCOLS,NROWS,V)
      CALL SOLVE(Z,V,Y,NCOLS,NROWS,NELEM,LM,MM,L,M,N,EXCITE)
      EXCITE=.FALSE.
      WRITE(3,300)
      READ(1,100) FLAG2
      IF(FLAG2.NE.0) GO TO 25
```

```
      CALL PATTRN(NCOLS,NROWS,NELEM,Y,DX,DZ)
      CALL SHAPE(V,NSIDE,NT)
      CALL PATTRN(NCOLS,NROWS,NELEM,V,DX,DZ)
   25 CONTINUE
      IF(FLAG.EQ.0) GO TO 30
      STOP
  100 FORMAT(2I0,5F0.0,I0)
  200 FORMAT(1H ,4E12.4)
  300 FORMAT(1H ,'TYPE 0 TO  PRINT ARRAY FACTOR, ELSE 1')
      END
C
C     ROUTINE TO COMPUTE THE FAR FIELD PATTERN
      SUBROUTINE PATTRN(LO,MO,NO,Y,DX,DZ)
      INTEGER LO,MO,NO,PTR
      REAL DX,DZ,ALPHA,COSTHE,MAGE,DANG,T1,T2,T3,DEL
      COMPLEX Y(1),AF,XPHASE,ZPHASE,DXPH,DZPH,CMPLX
      REAL PAT(361)
      WRITE(3,1000)
      READ(1,1100) NPTS,DANG
      ALPHA=0.0
      DO 300 II=1,NPTS
      COSTHE=COS(ALPHA*.17453293E-01)
      T3=DX*COSTHE
      T1=COS(T3)
      T2=SIN(T3)
      DXPH=CMPLX(T1,T2)
      DZPH=DXPH
      AF=(0.0,0.0)
      ZPHASE=(1.0,0.0)
      DO 200 I=1,MO
      PTR=(I-1)*LO
      XPHASE=(1.0,0.0)
      DO 100 J=1,LO
      AF=AF+Y(PTR+J)*XPHASE*ZPHASE
      XPHASE=XPHASE*DXPH
  100 CONTINUE
      ZPHASE=ZPHASE*DZPH
  200 CONTINUE
      MAGE=CABS(AF)
      PAT(II)=MAGE
      WRITE(3,1200) ALPHA,MAGE,AF
      ALPHA=ALPHA+DANG
  300 CONTINUE
      WRITE(21,1300) (PAT(II),II=1,NPTS)
 1000 FORMAT(1H ,'NPTS AND ANGLE INCREMENT?')
 1100 FORMAT(I0,E0.0)
 1200 FORMAT(1H ,F8.1,3E12.4)
 1300 FORMAT(8E11.4)
      RETURN
      END
C
C     ROUTINE TO ZEROIZE CORNERS
      SUBROUTINE SHAPE(Y,NS,LO)
      COMPLEX Y(1),CZERO
      CZERO=(0.0,0.0)
      DO 30 I=1,NS-1
      IPTR=(I-1)*LO
      DO 10 J=1,NS-I
      Y(IPTR+J)=CZERO
   10 CONTINUE
```

```
      DO 20 J=2*NS+I-1,3*NS-2
      Y(IPTR+J)=CZERO
   20 CONTINUE
   30 CONTINUE
      DO 60 I=2*NS,3*NS-2
      IPTR=(I-1)*LO
      DO 40 J=1,I-2*NS+1
      Y(IPTR+J)=CZERO
   40 CONTINUE
      DO 50 J=5*NS-I-2,3*NS-2
      Y(IPTR+J)=CZERO
   50 CONTINUE
   60 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SOLVE(A,B,LO,MO,NO,LM,MM,L,M,N,FXCITE)
C     ROUTINE TO SOLVE THE MATRIX EQUATION A X = B
      LOGICAL FXCITE
      COMPLEX CTEMP,CZERO,A(1),X(4096),V(4096),B(1),Y(7396)
      INTEGER FLAG1,FLAG2,COUNT
      REAL CHNAVG,CHNMAX,CHANGE
      CZERO=(0.0,0.0)
C     ZEROIZE Y AND V (EXPENDED B)
      DO 10 I=1,N
      V(I)=CZERO
   10 CONTINUE
      DO 20 I=1,NO
      Y(I)=CZERO
   20 CONTINUE
      IF(.NOT.FXCITE) GO TO 65
      DO 40 I=1,MO
      IPTR=(I-1)*L
      JPTR=IPTR+LO+LO-1
      DO 30 J=IPTR+1,IPTR+LO-1
      A(JPTR)=A(J)
      JPTR=JPTR-1
   30 CONTINUE
   40 CONTINUE
C     FILL UP A ARRAY AND V ARRAY
      DO 60 I=1,MO-1
      IPTR=(I-1)*L
      JPTR=(MO+MO-I-1)*L
      DO 50 J=1,LO+LO-1
      A(JPTR+J)=A(IPTR+J)
   50 CONTINUE
   60 CONTINUE
      CALL TWODF(A,N,M,L,MM,LM)
   65 CONTINUE
      CALL ITER(A,B,X,V,Y,LO,MO,NO,LM,MM,L,M,N)
      RETURN
      END
      SUBROUTINE ITER(A,B,X,V,Y,LO,MO,NO,LM,MM,L,M,N)
      INTEGER COUNT,FLAG1,FLAG2
      COMPLEX CZERO,CTEMP,A(1),B(1),X(1),V(1),Y(1)
      REAL CHNAVG,CHNMAX,CHANGE
      COUNT=0
      CZERO=(0.0,0.0)
   70 JPTR=1
      COUNT=COUNT+1
      DO 90 I=1,MO
      IPTR=L*(MO-2+I)+LO
      DO 80 J=1,LO
      V(IPTR)=B(JPTR)
      JPTR=JPTR+1
      IPTR=IPTR+1
   80 CONTINUE
   90 CONTINUE
C     FIND V TRANSFORMED AND COMPUTE X TRANSFORMED
      CALL TWODF(V,N,M,L,MM,LM)
      DO 100 I=1,N
      X(I)=V(I)/A(I)
  100 CONTINUE
C     GET X FROM X TRANSFORMED
      CALL ITWODF(X,N,M,L,MM,LM)
C     TRUNCATE X AND SAVE X AFTER COMPUTING THE CONVERGENCE CRITERION
```

```
      CHNAVG=0.0
      CHNMAX=0.0
      DO 120 I=1,N
      IPTR=I/L
      JPTR=I-IPTR*L
      IF(JPTR.LE.LO.AND.JPTR.NE.0.AND.IPTR.LT.MO) GO TO 110
      X(I)=CZERO
      GO TO 120
  110 IPTR=IPTR*LO+JPTR
      CTEMP=X(I)
      CHANGE=CABS(CTEMP-Y(IPTR))/CABS(CTEMP)
      Y(IPTR)=CTEMP
      CHNAVG=CHNAVG+CHANGE
      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
  120 CONTINUE
      CHNAVG=(CHNAVG*100.0)/FLOAT(NO)
      CHNMAX=CHNMAX*100.0
      WRITE(3,1200) CHNAVG,CHNMAX,COUNT
C     FIND THE TRANSFORM OF TRUNCATED X
      CALL TWODF(X,N,M,L,MM,LM)
C     COMPUTE V TRANSFORMED
      DO 130 I=1,N
      V(I)=A(I)*X(I)
  130 CONTINUE
C     GET V FROM V TRANSFORMED
      CALL ITWODF(V,N,M,L,MM,LM)
C     COMPUTE THE ERROR CRITERION
      CHNAVG=0.0
      CHNMAX=0.0
      JPTR=1
      DO 150 I=1,MO
      IPTR=L*(MO-2+I)+LO
      DO 140 J=1,LO
      CTEMP=B(JPTR)
      CHANGE=CABS(CTEMP-V(IPTR))/CABS(CTEMP)
      CHNAVG=CHNAVG+CHANGE
      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
      IPTR=IPTR+1
      JPTR=JPTR+1
  140 CONTINUE
  150 CONTINUE
C     ASK WHETHER OR NOT TO STOP AFTER REPORTING % FIELD ERROR
      CHNAVG=(CHNAVG*100.0)/FLOAT(NO)
      CHNMAX=CHNMAX*100.0
      WRITE(3,1300)CHNMAX,CHNAVG
      READ(1,1100)FLAG1
      IF(FLAG1.EQ.0) GO TO 70
      WRITE(3,1400) (Y(I),I=1,NO)
C     ASK IF FIELD SHOULD BE PRINTED OUT ALSO
      WRITE(3,1500)
      READ(1,1100)FLAG2
      IF(FLAG2.NE.0)RETURN
      WRITE(3,1600) (V(I),I=1,N)
      RETURN
 1000 FORMAT(10E0.0)
 1100 FORMAT(4I0)
 1200 FORMAT(1H ,'AVG CURRENT CHANGE=',E14.7,' %'/
     $       1H ,'MAX CURRENT CHANGE=',E14.7,' %'/
     $       1H ,'AFTER',I4,' ITERATIONS'/)
 1300 FORMAT(1H ,'MAX FIELD ERROR = ',E15.7,' %'/
```

```
          $       1H ,'AVG FIELD ERROR = ',E15.7,' %'/
          $       1H ,'CONTINUE ITERATIONS? 0 FOR YES, 1 FOR NO, AND RETURN'/)
     1400   FORMAT(1H ,'CURRENTS'//(1H ,10E11.4))
     1500   FORMAT(1H ,'PRINT FIELDS? 0 FOR YES, 1 FOR NO, THEN RETURN'/)
     1600   FORMAT(1H ,'RESULTANT FIELDS'//(1H ,10E11.4))
            END
```

III. MAIN PROGRAM SEGMENT AND SUBROUTINES FOR THE PLANAR ARRAY PROBLEMS (THREE EXPANSION FUNCTIONS PER ANTENNA ELEMENT)

The difference between the main program segment for three expansion functions per antenna element and the main program segment for one expansion function per antenna element is that there are five distinct mutual impedance vectors $\vec{Z}$ to be computed for the three expansion functions per antenna element solution. Therefore the computation of each $\vec{Z}$ is done by the seperate routine FILLZ. Similarly, MSOLVE and SOLVE routines differ mainly in that there are three distinct vectors each for the generalized voltage $\vec{V}$ and $\vec{J}$ to be computed by MSOLVE. SOLVE on the other hand, computes only one vector each of $\vec{V}$ and $\vec{J}$.

Routine ZMNG computes the mutual impedance between two parallel segments of thin wire dipoles of same length which may or may not be offset from each other along one or more axes. SICI, VOLTU, VOLTC TAP, VOLTK, and PHASE are all from [9] and hence no description of them will be given here. FFT, IFFT, TWODF, and ITWODF are fast fourier transform and inverse transform routines for one and two dimensional discrete fourier transforms, respectively. PATT3E is the routine to compute the array factor from the current distribution solutions obtained by using three expansion functions per antenna element.

The last main program segment computes the array factors from the current distribution solutions obtained by using three expansion functions per antenna element and from the ideal solutions which ignore the mutual coupling between antenna elements.

```
C
C       PROGRAM FOR TRIANGULAR PATTERN WITH THREE EXPANSIONS PER
C       ELEMENT
        LOGICAL EXCITE
        INTEGER CODE,FLAG,NROWS,NCOLS,NELEM
        REAL TWOPI
        COMPLEX V2(484),ZA(4096),ZB(4096),ZC(4096),ZD(4096),ZE(4096)
        COMPLEX X1(4096),X2(4096),X3(4096)
        EXCITE=.TRUE.
        TWOPI=6.2831853
        CZERO=(0.0,0.0)
        OPEN(UNIT=21,FILE='MXCUR.DAT')
        READ(1,100) NSIDE,DX,DY,DZ,WLNGTH,RAD
        NROWS=NSIDE*3-2
        NCOLS=NROWS
        CALL SIZE(NCOLS,NROWS,LM,MM,L,M,N)
        DX=DX*TWOPI
        DY=DY*TWOPI
        DZ=DZ*TWOPI
        WLBY2=WLNGTH*TWOPI/4.0
        ZELEM=WLBY2
        RAD=RAD*TWOPI
        NELEM=NROWS*NCOLS
        DY2SQ=4.0*DY*DY
        RAD2=RAD*RAD
        ZOFF=0.0
        CALL FILLZ(ZA,ZOFF,DX,DZ,DY2SQ,WLBY2,RAD2,NROWS,L,N)
        ZOFF=-ZELEM
        CALL FILLZ(ZB,ZOFF,DX,DZ,DY2SQ,WLBY2,RAD2,NROWS,L,N)
        ZOFF=-ZELEM-ZELEM
        CALL FILLZ(ZC,ZOFF,DX,DZ,DY2SQ,WLBY2,RAD2,NROWS,L,N)
        CALL FILLZ(ZD,ZELEM,DX,DZ,DY2SQ,WLBY2,RAD2,NROWS,L,N)
        ZOFF=ZELEM+ZELEM
        CALL FILLZ(ZE,ZOFF,DX,DZ,DY2SQ,WLBY2,RAD2,NROWS,L,N)
   10   READ(1,200) CODE,FLAG
        IF(CODE.EQ.1) CALL VOLTU(NCOLS,NROWS,V2)
        IF(CODE.EQ.2) CALL VOLTC(NCOLS,NROWS,V2)
        IF(CODE.EQ.3) CALL TAP(NCOLS,NROWS,DX,DZ,V2)
        IF(CODE.EQ.4) CALL VOLTK(NCOLS,NROWS,V2)
        IF(CODE.EQ.5) CALL STEER(NCOLS,NROWS,DX,DZ,V2)
        IF(CODE.EQ.6) CALL PHASE(NCOLS,NROWS,V2)
        CALL MSOLVE(ZA,ZB,ZC,ZD,ZE,V2,X1,X2,X3,NCOLS,NROWS,
     $              NELEM,LM,MM,L,M,N,EXCITE)
        EXCITE=.FALSE.
        IF(FLAG.EQ.0) GO TO 10
        STOP
  100   FORMAT(I0,5F0.0,I0)
  200   FORMAT(2I0)
        END
C
C       ROUTINE TO FILL THE Z MATRIX FOR TRIANGULAR ARRAYS
        SUBROUTINE FILLZ(Z,ZOFF,DX,DZ,DY2SQ,WLBY2,RAD2,NROWS,L,N)
        INTEGER POSPTR,NROWS,L,N
        REAL DX,DZ,DY2SQ,ZELEM,XCUR,ZCUR,RR,WLBY2
        COMPLEX Z(1),MUTUAL,IMAGE,CZERO,ZMNG
        CZERO=(0.0,0.0)
        ZELEM=0.0
        POSPTR=1
        DO 10 I=1,N
        Z(I)=CZERO
```

```
   10 CONTINUE
      NTOTAL=NROWS+NCOLS-1
      DO 30 I=1,NTOTAL
      XCUR=(I-1)*DX
      ZCUR=(NTOTAL-I)*DZ+ZOFF
      DO 20 J=1,NTOTAL
      RR=XCUR*XCUR+DY2SQ
      IMAGE=ZMNG(ZELEM,ZCUR,WLBY2,RR)
      RR=RR-DY2SQ
      IF(RR.EQ.0.0) RR=RR+RAD2
      MUTUAL=ZMNG(ZELEM,ZCUR,WLBY2,RR)
      Z(POSPTR)=MUTUAL-IMAGE
      POSPTR=POSPTR+1
      XCUR=XCUR-DX
      ZCUR=ZCUR-DZ
   20 CONTINUE
      POSPTR=POSPTR+L-NTOTAL
   30 CONTINUE
      RETURN
      END
C
C     ROUTINE TO SOLVE THE CONVOLUTION EQUATIONS
      SUBROUTINE MSOLVE(ZA,ZB,ZC,ZD,ZE,B,X1,X2,X3,LO,MO,NO,
     $                  LM,MM,L,M,N,EXCITE)
      INTEGER COUNT
      LOGICAL EXCITE
      COMPLEX CTEMP1,CTEMP2,CTEMP3,ZA(1),ZB(1),ZC(1),ZD(1),ZE(1),
     $        B(1),X1(1),X2(1),X3(1),V1(4096),V2(4096),V3(4096),
     $        T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,Y1(484),Y2(484),Y3(484),
     $        CZERO
      CZERO=(0.0,0.0)
      DO 10 I=1,NO
      Y1(I)=CZERO
      Y2(I)=CZERO
      Y3(I)=CZERO
   10 CONTINUE
      NS=(LO+2)/3
      EFACT=3.0*FLOAT(NO-2*(NS-1)*NS)
      DO 20 I=1,N
      V1(I)=CZERO
      V2(I)=CZERO
      V3(I)=CZERO
   20 CONTINUE
      IF(.NOT.EXCITE) GO TO 30
      CALL TWODF(ZA,N,M,L,MM,LM)
      CALL TWODF(ZB,N,M,L,MM,LM)
      CALL TWODF(ZC,N,M,L,MM,LM)
      CALL TWODF(ZD,N,M,L,MM,LM)
      CALL TWODF(ZE,N,M,L,MM,LM)
   30 ICENTR=L*(MO-1)+LO-1
      COUNT=1
   40 CONTINUE
      DO 60 I=1,NS-1
      IPTR=(I-1)*L+ICENTR
      JPTR=(I-1)*LO
      DO 50 J=NS-I+1,2*NS+I-2
      V1(IPTR+J)=CZERO
      V2(IPTR+J)=B(JPTR+J)
      V3(IPTR+J)=CZERO
   50 CONTINUE
```

```
   60 CONTINUE
      DO 80 I=NS,2*NS-1
      IPTR=(I-1)*L+ICENTR
      JPTR=(I-1)*LO
      DO 70 J=1,3*NS-2
      V1(IPTR+J)=CZERO
      V2(IPTR+J)=B(JPTR+J)
      V3(IPTR+J)=CZERO
   70 CONTINUE
   80 CONTINUE
      DO 100 I=2*NS,3*NS-2
      IPTR=(I-1)*L+ICENTR
      JPTR=(I-1)*LO
      DO 90 J=I+2-2*NS,5*NS-I-3
      V1(IPTR+J)=CZERO
      V2(IPTR+J)=B(JPTR+J)
      V3(IPTR+J)=CZERO
   90 CONTINUE
  100 CONTINUE
  105 CONTINUE
C     FIND V1,V2,V3 TRANSFORMED AND COMPUTE X1,X2,X3
      CALL TWODF(V1,N,M,L,MM,LM)
      CALL TWODF(V2,N,M,L,MM,LM)
      CALL TWODF(V3,N,M,L,MM,LM)
      DO 110 I=1,N
      T1=ZD(I)/ZA(I)
      T2=ZE(I)/ZA(I)
      T3=ZA(I)-T1*ZB(I)
      T4=ZB(I)-T1*ZC(I)
      T5=ZA(I)-T2*ZC(I)
      T6=ZD(I)-T2*ZB(I)
      T7=V2(I)-V1(I)*T1
      T8=V3(I)-V1(I)*T2
      T9=T5-T4*T6/T3
      T10=T8-T7*T6/T3
      X3(I)=T10/T9
      X2(I)=(T7-T4*X3(I))/T3
      X1(I)=(V1(I)-ZB(I)*X2(I)-ZC(I)*X3(I))/ZA(I)
  110 CONTINUE
      CALL ITWODF(X1,N,M,L,MM,LM)
      CALL ITWODF(X2,N,M,L,MM,LM)
      CALL ITWODF(X3,N,M,L,MM,LM)
      DO 140 I=1,NS-1
      IPTR=(I-1)*L
      DO 120 J=1,NS-I
      X1(IPTR+J)=CZERO
      X2(IPTR+J)=CZERO
      X3(IPTR+J)=CZERO
  120 CONTINUE
      DO 130 J=2*NS+I-1,3*NS-2
      X1(IPTR+J)=CZERO
      X2(IPTR+J)=CZERO
      X3(IPTR+J)=CZERO
  130 CONTINUE
  140 CONTINUE
      DO 170 I=2*NS,3*NS-2
      IPTR=(I-1)*L
      DO 150 J=1,I-2*NS+1
      X1(IPTR+J)=CZERO
      X2(IPTR+J)=CZERO
```

```
      X3(IPTR+J)=CZERO
  150 CONTINUE
      DO 160 J=5*NS-I-2,3*NS-2
      X1(IPTR+J)=CZERO
      X2(IPTR+J)=CZERO
      X3(IPTR+J)=CZERO
  160 CONTINUE
  170 CONTINUE
C
C     COMPUTE THE CRITERIONS AND REPORT
      CHNAVG=0.0
      CHNMAX=0.0
      DO 190 I=1,N
      IPTR=I/L
      JPTR=I-IPTR*L
      IF(JPTR.LE.LO.AND.JPTR.NE.0.AND.IPTR.LT.MC) GO TO 180
      X1(I)=CZERO
      X2(I)=CZERO
      X3(I)=CZERO
      GO TO 190
  180 CONTINUE
      IPTR=IPTR*LO+JPTR
      CTEMP1=X1(I)
      CTEMP2=X2(I)
      CTEMP3=X3(I)
      IF(CTEMP1.EQ.CZERO) GO TO 190
      CHNGE1=CABS(CTEMP1-Y1(IPTR))/CABS(CTEMP1)
      CHNGE2=CABS(CTEMP2-Y2(IPTR))/CABS(CTEMP2)
      CHNGE3=CABS(CTEMP3-Y3(IPTR))/CABS(CTEMP3)
      Y1(IPTR)=CTEMP1
      Y2(IPTR)=CTEMP2
      Y3(IPTR)=CTEMP3
      CHNAVG=CHNAVG+CHNGE1+CHNGE2+CHNGE3
      IF(CHNGE1.GT.CHNMAX) CHNMAX=CHNGE1
      IF(CHNGE2.GT.CHNMAX) CHNMAX=CHNGE2
      IF(CHNGE3.GT.CHNMAX) CHNMAX=CHNGE3
  190 CONTINUE
      CHNAVG=(CHNAVG*100.0)/EFACT
      CHNMAX=(CHNMAX*100.0)
      WRITE(3,1200) CHNAVG,CHNMAX,COUNT
C
C     GET THE  TRANSFORM OF THE CURRENTS
      CALL TWODF(X1,N,M,L,MM,LM)
      CALL TWODF(X2,N,M,L,MM,LM)
      CALL TWODF(X3,N,M,L,MM,LM)
      DO 200 I=1,N
      V1(I)=ZA(I)*X1(I)+ZB(I)*X2(I)+ZC(I)*X3(I)
      V2(I)=ZD(I)*X1(I)+ZA(I)*X2(I)+ZB(I)*X3(I)
      V3(I)=ZE(I)*X1(I)+ZD(I)*X2(I)+ZA(I)*X3(I)
  200 CONTINUE
C
C     GET THE FIELD AND COMPUTE % ERRORS
      CALL ITWODF(V1,N,M,L,MM,LM)
      CALL ITWODF(V2,N,M,L,MM,LM)
      CALL ITWODF(V3,N,M,L,MM,LM)
      CHNAVG=0.0
      CHNMAX=0.0
      DO 220 I=1,NS-1
      IPTR=(I-1)*L+ICENTR
      JPTR=(I-1)*LO
```

```
      DO 210 J=NS-I+1,2*NS+I-2
      CTEMP1=B(JPTR+J)
      CHANGE=CABS(CTEMP1-V2(IPTR+J))/CABS(CTEMP1)
      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
      CHNAVG=CHNAVG+CHANGE
      V2(IPTR+J)=CTEMP1
      V1(IPTR+J)=CZERO
      V3(IPTR+J)=CZERO
  210 CONTINUE
  220 CONTINUE
      DO 240 I=NS,NS*2-1
      IPTR=(I-1)*L+ICENTR
      JPTR=(I-1)*LO
      DO 230 J=1,3*NS-2
      CTEMP1=B(JPTR+J)
      CHANGE=CABS(CTEMP1-V2(IPTR+J))/CABS(CTEMP1)
      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
      CHNAVG=CHNAVG+CHANGE
      V1(IPTR+J)=CZERO
      V2(IPTR+J)=CTEMP1
      V3(IPTR+J)=CZERO
  230 CONTINUE
  240 CONTINUE
      DO 260 I=2*NS,3*NS-2
      IPTR=(I-1)*L+ICENTR
      JPTR=(I-1)*LO
      DO 250 J=I+2-2*NS,5*NS-I-3
      CTEMP1=B(JPTR+J)
      CHANGE=CABS(CTEMP1-V2(IPTR+J))/CABS(CTEMP1)
      IF(CHANGE.GT.CHNMAX) CHNMAX=CHANGE
      CHNAVG=CHNAVG+CHANGE
      V2(IPTR+J)=CTEMP1
      V1(IPTR+J)=CZERO
      V3(IPTR+J)=CZERO
  250 CONTINUE
  260 CONTINUE
C     ASK WHETHER OR NOT TO STOP AFTER REPORTING % FIELD ERROR
      CHNAVG=(CHNAVG*100.0)/EFACT
      CHNMAX=CHNMAX*100.0
      WRITE(3,1300)CHNAVG,CHNMAX
      READ(1,1100)FLAG1
      COUNT=COUNT+1
      IF(FLAG1.EQ.0) GO TO 105
      WRITE(3,1400) (Y1(I),I=1,NO)
      WRITE(21,1700) (Y1(I),I=1,NO)
      WRITE(3,1400) (Y2(I),I=1,NO)
      WRITE(21,1700) (Y2(I),I=1,NO)
      WRITE(3,1400) (Y3(I),I=1,NO)
      WRITE(21,1700) (Y3(I),I=1,NO)
C     ASK IF FIELD SHOULD BE PRINTED OUT ALSO
      WRITE(3,1500)
      READ(1,1100) FLAG2
      IF(FLAG2.NE.0) RETURN
      WRITE(3,1600) (V2(I),I=1,N)
      RETURN
 1000 FORMAT(10E0.0)
 1100 FORMAT(4I0)
 1200 FORMAT(1H ,'AVG CURRENT CHANGE=',E14.7,' %'/
     $       1H ,'MAX CURRENT CHANGE=',E14.7,' %'//
     $       1H ,'AFTER',I4,' ITERATIONS'/)
```

```
 1300   FORMAT(1H ,'AVG FIELD ERROR = ',E15.7,' %'/
     $      1H ,'MAX FIELD ERROR = ',E15.7,' %'/
     $      1H ,'CONTINUE ITERATIONS? 0 FOR YES, 1 FOR NO, AND RETURN'/)
 1400   FORMAT(1H ,'CURRENTS'//(1H ,10E11.4))
 1500   FORMAT(1H ,'PRINT FIELDS? 0 FOR YES, 1 FOR NO, THEN RETURN'/)
 1600   FORMAT(1H ,'RESULTANT FIELDS'//(1H ,10E11.4))
 1700   FORMAT(8E14.6)
        END
C
C       COMPUTE MUTUAL IMPEDANCE BETWEEN TWO PARALLEL SEGMENTS
C       OF THIN WIRE DIPOLES OF SAME LENGTH
        FUNCTION ZMNG(Z1,Z2,LBY2,RSQ)
        REAL LBY2
        COMPLEX ZMNG,CMPLX
        DZ=ABS(Z1-Z2)
        CC=2.0*COS(LBY2)
        CSQ=CC*CC
        D1=DZ
        D2=DZ+LBY2
        D3=DZ-LBY2
        D4=D2+LBY2
        D5=D3-LBY2
        U1=SQRT(RSQ+D1*D1)+D1
        U2=SQRT(RSQ+D2*D2)+D2
        U3=SQRT(RSQ+D3*D3)+D3
        U4=SQRT(RSQ+D4*D4)+D4
        U5=SQRT(RSQ+D5*D5)+D5
        V1=RSQ/U1
        V2=RSQ/U2
        V3=RSQ/U3
        V4=RSQ/U4
        V5=RSQ/U5
        CALL SICI(SU1,CU1,U1)
        CALL SICI(SU2,CU2,U2)
        CALL SICI(SU3,CU3,U3)
        CALL SICI(SU4,CU4,U4)
        CALL SICI(SU5,CU5,U5)
        CALL SICI(SV1,CV1,V1)
        CALL SICI(SV2,CV2,V2)
        CALL SICI(SV3,CV3,V3)
        CALL SICI(SV4,CV4,V4)
        CALL SICI(SV5,CV5,V5)
        S1=SIN(D1)
        S2=SIN(D2)
        S3=SIN(D3)
        S4=SIN(D4)
        S5=SIN(D5)
        C1=COS(D1)
        C2=COS(D2)
        C3=COS(D3)
        C4=COS(D4)
        C5=COS(D5)
        RL=(2.0+CSQ)*(C1*(CU1+CV1)+S1*(SU1-SV1))
     $    -2.0*CC*(C2*(CU2+CV2)+S2*(SU2-SV2)+C3*(CU3+CV3)
     $    +S3*(SU3-SV3))+C4*(CU4+CV4)+S4*(SU4-SV4)
     $    +C5*(CU5+CV5)+S5*(SU5-SV5)
        AG=(2.0+CSQ)*(S1*(CU1-CV1)-C1*(SU1+SV1))
     $    -2.0*CC*(S2*(CU2-CV2)-C2*(SU2+SV2)+S3*(CU3-CV3)
     $    -C3*(SU3+SV3))+S4*(CU4-CV4)-C4*(SU4+SV4)
     $    +S5*(CU5-CV5)-C5*(SU5+SV5)
```

```
      ZMNG=15.0*CMPLX(RL,AG)/(SIN(LBY2)*SIN(LBY2))
      RETURN
      END
C
C
      SUBROUTINE SICI(SI,CI,X)
      Z=ABS(X)
      IF(Z-4.) 1,1,4
    1 Y=(4.-Z)*(4.+Z)
    3 SI=X*(((((1.753141E-9*Y+1.568988E-7)*Y+1.374168E-5)*Y+6.939889E-4)
     1*Y+1.964882E-2)*Y+4.395509E-1)
      CI=((5.772156E-1+ALOG(Z))/Z-Z*(((((1.386985E-10*Y+1.584996E-8)*Y
     1+1.725752E-6)*Y+1.185999E-4)*Y+4.990920E-3)*Y+1.315308E-1))*Z
      RETURN
    4 SI=SIN(Z)
      Y=COS(Z)
      Z=4./Z
      U=((((((((4.048069E-3*Z-2.279143E-2)*Z+5.515070E-2)*Z-7.261642E-2)
     1*Z+4.987716E-2)*Z-3.332519E-3)*Z-2.314617E-2)*Z-1.134958E-5)*Z
     2+6.250011E-2)*Z+2.583989E-10
      V=(((((((((-5.108699E-3*Z+2.819179E-2)*Z-6.537283E-2)*Z
     1+7.902034E-2)*Z-4.400416E-2)*Z-7.945556E-3)*Z+2.601293E-2)*Z
     2-3.764000E-4)*Z-3.122418E-2)*Z-6.646441E-7)*Z+2.500000E-1
      CI=Z*(SI*V-Y*U)
      SI=-Z*(SI*U+Y*V)      +1.570796
      RETURN
      END
C
C
C
      SUBROUTINE  VOLTU(M2,M3,V)
C     UNIFORM EXCITATION WITH SPECIFIED AMPLITUDE AND PHASE
      COMPLEX V(1),CMPLX,VALUE
      M23=M2*M3
      READ (5,1)AM,PH
 1    FORMAT(2F0.0)
      RAD=PH*3.14159/180.
      VALUE=CMPLX(AM*COS(RAD),AM*SIN(RAD))
      DO 2 I=1,M23
      V(I)=VALUE
 2    CONTINUE
      WRITE(5,3)AM,PH
 3    FORMAT(///1X,'UNIFORM VOLTAGE EXCITATION -'//1X,
     &'MAGNITUDE =',1P1E20.5,'    PHASE =',1P1E20.5/)
      RETURN
      END
C
C
C
      SUBROUTINE VOLTC(M2,M3,V)
C     READ IN COMPLEX NUMBERS AS VOLTAGE FOR EACH DIPOLE
      COMPLEX V(1)
      M23=M2*M3
      READ (5,1) (V(I),I=1,M23)
 1    FORMAT(2F0.0)
      WRITE(5,2)
 2    FORMAT(///1X,'ARBITRARY VOLTAGE EXCITATION -'//)
      RETURN
      END
C
```

```
C
      SUBROUTINE STEER(M2,M3,DX,DZ,PP)
C     PROGRESSIVE PHASE SHIFT ON EACH DIPOLE -
C     STEERING THE MAIN BEAM IN BOTH DIRECTION
      COMPLEX PP(1),CMPLX
      READ (5,1) RZ,RX
 1    FORMAT(2F0.0)
      PI2=6.2831853
      THX=(RX)*PI2/360.
      THZ=(RZ)*PI2/360.
      L=0
      XK=-DX*COS(THX)
      ZK=-DZ*COS(THZ)
      DO 100 I=1,M3
      DO 100 J=1,M2
      L=L+1
      PH=FLOAT(J-1)*XK+FLOAT(I-1)*ZK
      PP(L)=1.0*CMPLX(COS(PH),SIN(PH))
  100 CONTINUE
      WRITE(5,20)RX,RZ
 20   FORMAT(///1X,'BEAM STEERING =',F10.5,' DEGREES IN PHI ANGLE',
     &/1X,14X,'=',F10.5,' DEGREES IN THE ANGLE'//)
      RETURN
      END
C
C
      SUBROUTINE TAP(M2,M3,DX,DZ,VV)
C     MAGNITUDE TAPER OF EXCITATION IN BOTH DIRECTION
      COMPLEX VV(1)
      PI2=6.2831853
      HFX=DX*(M2-1)*0.5
      HFZ=DZ*(M3-1)*0.5
      L=0
      WRITE(5,2)
 2    FORMAT(///1X,'EXPONENTIAL TAPERED IN MAGNITUDE'///)
      DO 100 I=1,M3
      Z=((I-1)*DZ-HFZ)/PI2
      FUNZ=EXP(-ABS(Z))
      DO 100 J=1,M2
      L=L+1
      X=((J-1)*DX-HFX)/PI2
      VV(L)=EXP(-ABS(X))*FUNZ
 100  CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE VOLTK(M2,M3,V)
      COMPLEX V(1),VK(20),VJ(20)
      READ (5,1) (VK(I),I=1,M3)
      READ (5,1) (VJ(I),I=1,M2)
 1    FORMAT(10F0.0)
      L=0
      DO 2 I=1,M3
      DO 2 J=1,M2
      L=L+1
      V(L)=VK(I)*VJ(J)
 2    CONTINUE
      WRITE(5,4)
```

```
 4    FORMAT(///1X,'VOLTAGE EXCITATION - SPECIFIED BY ROW',
     &' AND COLUMN',//)
      RETURN
      END
C
C
      SUBROUTINE PHASE(M2,M3,PP)
C     PROGRESSIVE PHASE SHIFT ON EACH DIPOLE -
      COMPLEX PP(1),CMPLX
      READ (5,1) RZ,RX
 1    FORMAT(2F0.0)
      PI2=6.2831853
      THX=(RX)*PI2/360.
      THZ=(RZ)*PI2/360.
      L=0
      DO 100 I=1,M3
      DO 100 J=1,M2
      L=I+1
      PH=FLOAT(J-1)*THX+FLOAT(I-1)*THZ
      PP(L)=1.0*CMPLX(COS(PH),SIN(PH))
  100 CONTINUE
      WRITE(5,20) RX,RZ
 20   FORMAT(///1X,'PROGRESSIVE PHASE SHIFT = ',F10.5,
     &'DEGREES IN ROW DIRECTION'/25X,'= ',F10.5,
     &'DEGREES IN COLUMN DIRECTION',///)
      RETURN
      END
       SUBROUTINE SIZE(LO,MO,LM,MM,L,M,N)
       L=LO*3-2
       M=MO*3-2
       LTEMP=L
       MTEMP=M
       LM=0
       MM=0
    1  L=L/2
       LM=LM+1
       IF(L.GT.1) GO TO 1
    2  M=M/2
       MM=MM+1
       IF(M.GT.1) GO TO 2
       L=2**LM
       M=2**MM
       IF(LTEMP.GT.L) LM=LM+1
       IF(LTEMP.GT.L) L=L*2
       IF(MTEMP.GT.M) MM=MM+1
       IF(MTEMP.GT.M) M=M*2
       N=M*L
       RETURN
       END
       SUBROUTINE FFT(X,M,START,STEP)
       COMPLEX X(16384),U,W,T
       INTEGER START,STEP,SDIFF
       N=2**M
       SDIFF=STEP-START
       NV2  =N/2*STEP
       NM1  =(N-2)*STEP+START
       N    =(N-1)*STEP+START
       J    =START
       DO 8 I=START,NM1,STEP
       IF(I.GE.J) GO TO 5
```

```
            T     =X(J)
            X(J)  =X(I)
            X(I)  =T
          5 K     =NV2
          6 IF(K-SDIFF.GE.J) GO TO 7
            J     =J-K
            K     =K/2
            GO TO 6
          7 J     =J+K
          8 CONTINUE
            PI    =3.14159265358979
            DO 20 L=1,M
            LE    =2**L
            LE1=LE/2
            LSTEP=LE1*STEP
            U     =(1.0,0.0)
            ANGLE=PI/FLOAT(LE1)
            W     =CMPLX(COS(ANGLE),-SIN(ANGLE))
            LE1   =LSTEP+START-STEP
            LE    =LE*STEP
            DO 20 J=START,LE1,STEP
            DO 10 I=J,N,LE
            IP    =I+LSTEP
            T     =X(IP)*U
            X(IP)=X(I)-T
            X(I)  =X(I)+T
         10 CONTINUE
            U=U*W
         20 CONTINUE
            RETURN
            END
      C
            SUBROUTINE TWODF(X,N,M,L,MM,LM)
            COMPLEX X(16384)
            INTEGER START,STEP
            START=1
            STEP =1
            DO 10 I=1,M
            CALL FFT(X,LM,START,STEP)
            START=START+L
         10 CONTINUE
            STEP =L
            DO 20 I=1,L
            START=I
            CALL FFT(X,MM,START,STEP)
         20 CONTINUE
            RETURN
            END
            SUBROUTINE IFFT(X,M,START,STEP)
            COMPLEX X(16384),U,W,T
            INTEGER START,STEP,SDIFF
            N=2**M
            SDIFF=STEP-START
            NV2  =N/2*STEP
            NM1  =(N-2)*STEP+START
            NEXP =(N-1)*STEP+START
            J     =START
            DO 8 I=START,NM1,STEP
            IF(I.GE.J) GO TO 5
            T     =X(J)
```

```
      X(J) =X(I)
      X(I) =T
    5 K    =NV2
    6 IF(K-SDIFF.GE.J) GO TO 7
      J    =J-K
      K    =K/2
      GO TO 6
    7 J    =J+K
    8 CONTINUE
      PI   =3.14159265358979
      DO 20 L=1,M
      LE   =2**L
      LE1=LE/2
      LSTEP=LE1*STEP
      U    =(1.0,0.0)
      ANGLE=PI/FLOAT(LE1)
      W    =CMPLX(COS(ANGLE),SIN(ANGLE))
      LE1  =LSTEP+START-STEP
      LE   =LE*STEP
      DO 20 J=START,LE1,STEP
      DO 10 I=J,NEXP,LE
      IP   =I+LSTEP
      T    =X(IP)*U
      X(IP)=X(I)-T
      X(I) =X(I)+T
   10 CONTINUE
      U=U*W
   20 CONTINUE
      RETURN
      END
C
      SUBROUTINE ITWODF(X,N,M,L,MM,LM)
      COMPLEX X(16384)
      INTEGER START,STEP
      START=1
      STEP =1
      DO 10 I=1,M
      CALL IFFT(X,LM,START,STEP)
      START=START+L
   10 CONTINUE
      STEP =L
      DO 20 I=1,L
      START=I
      CALL IFFT(X,MM,START,STEP)
   20 CONTINUE
      FN=FLOAT(N)
      DO 30 I=1,N
      X(I)=X(I)/FN
   30 CONTINUE
      RETURN
      END
```

```
C
C         ROUTINE TO COMPUTE THE FAR FIELD PATTERN
          SUBROUTINE PATTRN(LO,MO,NO,Y,DX,DZ)
          INTEGER LO,MO,NO,PTR
          REAL DX,DZ,ALPHA,COSTHE,MAGE,DANG,T1,T2,T3,T4,DEL
          COMPLEX Y(1),AF,XPHASE,ZPHASE,DXPH,DZPH,CMPLX
          REAL PAT(361)
          WRITE(3,1000)
          READ(1,1100) NPTS,DANG,IFLAG
          ALPHA=0.0
          DO 300 II=1,NPTS
          COSTHE=COS(ALPHA*.17453293E-01)
          T3=DX*COSTHE
          T1=COS(T3)
          T2=SIN(T3)
          T4=-T2
          DXPH=CMPLX(T1,T2)
          DZPH=DXPH
          AF=(0.0,0.0)
          ZPHASE=(1.0,0.0)
          IF(IFLAG.NE.0) DZPH=CMPLX(T1,T4)
          DO 200 I=1,MO
          PTR=(I-1)*LO
          XPHASE=(1.0,0.0)
          DO 100 J=1,LO
          AF=AF+Y(PTR+J)*XPHASE*ZPHASE
          XPHASE=XPHASE*DXPH
    100   CONTINUE
          ZPHASE=ZPHASE*DZPH
    200   CONTINUE
          MAGE=CABS(AF)
          PAT(II)=MAGE
          WRITE(3,1200) ALPHA,MAGE,AF
          ALPHA=ALPHA+DANG
    300   CONTINUE
          WRITE(21,1300) (PAT(II),II=1,NPTS)
   1000   FORMAT(1H ,'NPTS AND ANGLE INCREMENT?')
   1100   FORMAT(I0,E0.0,I0)
   1200   FORMAT(1H ,F8.1,3E12.4)
   1300   FORMAT(8E11.4)
          RETURN
          END
C
C         ROUTINE TO ZEROIZE CORNERS
          SUBROUTINE SHAPE(Y,NS,LO)
          COMPLEX Y(1),CZERO
          CZERO=(0.0,0.0)
          DO 30 I=1,NS-1
          IPTR=(I-1)*LO
          DO 10 J=1,NS-I
          Y(IPTR+J)=CZERO
     10   CONTINUE
          DO 20 J=2*NS+I-1,3*NS-2
          Y(IPTR+J)=CZERO
     20   CONTINUE
     30   CONTINUE
          DO 60 I=2*NS,3*NS-2
          IPTR=(I-1)*LO
          DO 40 J=1,I-2*NS+1
          Y(IPTR+J)=CZERO
```

```
   40 CONTINUE
      DO 50 J=5*NS-I-2,3*NS-2
      Y(IPTR+J)=CZERO
   50 CONTINUE
   60 CONTINUE
      RETURN
      END
C
C     ROUTINE TO COMPUTE THE FAR FIELD PATTERN(3 EXPANSIONS)
      SUBROUTINE PATT3E(LO,MO,NO,Y,DX,DZ,WLBY2)
      INTEGER LO,MO,NO,PTR,CPTR
      REAL DX,DZ,ALPHA,COSTHE,MAGE,DANG,T1,T2,T3,T4,DEL
      COMPLEX Y(1),AF1,AF2,AF3,XPHASE,ZPHASE,DXPH,DZPH,CMPLX,AF
      COMPLEX PHASE,FAC1,FAC3
      REAL PAT(361)
      WRITE(3,1000)
      READ(1,1100) NPTS,DANG,IFLAG
      ALPHA=0.0
      DO 300 II=1,NPTS
      COSTHE=COS(ALPHA*.17453293E-01)
      T3=DX*COSTHE
      T1=COS(T3)
      T2=SIN(T3)
      T4=-T2
      DXPH=CMPLX(T1,T2)
      DZPH=DXPH
      AF1=(0.0,0.0)
      AF2=(0.0,0.0)
      AF3=(0.0,0.0)
      ZPHASE=(1.0,0.0)
      IF(IFLAG.NE.0) DZPH=CMPLX(T1,T4)
      DO 200 I=1,MO
      PTR=(I-1)*LO
      XPHASE=(1.0,0.0)
      DO 100 J=1,LO
      PHASE=XPHASE*ZPHASE
      CPTR=PTR+J
      AF1=AF1+Y(CPTR)*PHASE
      CPTR=CPTR+NO
      AF2=AF2+Y(CPTR)*PHASE
      CPTR=CPTR+NO
      AF3=AF3+Y(CPTR)*PHASE
      XPHASE=XPHASE*DXPH
  100 CONTINUE
      ZPHASE=ZPHASE*DZPH
  200 CONTINUE
      AF=AF1+AF2+AF3
      T3=WLBY2/2.0*COSTHE
      T1=COS(T3)
      T2=SIN(T3)
      FAC1=CMPLX(T1,T2)
      T2=-T2
      FAC3=CMPLX(T1,T2)
      IF(IFLAG.EQ.0) GO TO 250
      AF=AF1*FAC1+AF2+AF3*FAC3
      IF(COSTHE.GE.0.99999) GO TO 250
      AF=AF*(T1-COS(WLBY2/2.0))/(COS(WLBY2*COSTHE)-COS(WLBY2))
  250 CONTINUE
      MAGE=CABS(AF)
      PAT(II)=MAGE
```

```
            WRITE(3,1200) ALPHA,MAGE,AF
            ALPHA=ALPHA+DANG
  300       CONTINUE
            WRITE(21,1300) (PAT(II),II=1,NPTS)
 1000       FORMAT(1H ,'NPTS AND ANGLE INCREMENT?')
 1100       FORMAT(I0,E0.0,I0)
 1200       FORMAT(1H ,F8.1,3E12.4)
 1300       FORMAT(8E11.4)
            RETURN
            END
C
C           MAIN PROGRAM
            COMPLEX V(12300)
            INTEGER CODE,FLAG,LO,MO,NO,SKIP
            OPEN(UNIT=21,FILE='PATTRN.DAT')
            OPEN(UNIT=22,FILE='MXCUR.DAT')
            READ(1,200) LO,MO,NO,DX,DZ,WLBY2,SKIP
   10       IF(SKIP.LE.0) GO TO 20
            READ(22,300) (V(I),I=1,NO)
            READ(22,300) (V(I),I=1,NO)
            READ(22,300) (V(I),I=1,NO)
            SKIP=SKIP-1
            GO TO 10
   20       CONTINUE
            TWOPI=6.2831853
            DX=DX*TWOPI
            DZ=DZ*TWOPI
            WLBY2=WLBY2*TWOPI
            READ(22,300) (V(I),I=1,NO)
            READ(22,300) (V(I),I=NO+1,NO+NO)
            READ(22,300) (V(I),I=NO+NO+1,NO+NO+NO)
            CALL PATT3E(LO,MO,NO,V,DX,DZ,WLBY2)
            NSIDE=(LO+2)/3
            NCOLS=LO
            NROWS=MO
            READ(1,200) CODE,FLAG
            IF(CODE.EQ.1) CALL VOLTU(NCOLS,NROWS,V)
            IF(CODE.EQ.2) CALL VOLTC(NCOLS,NROWS,V)
            IF(CODE.EQ.3) CALL TAP(NCOLS,NROWS,DX,DZ,V)
            IF(CODE.EQ.4) CALL VOLTK(NCOLS,NROWS,V)
            IF(CODE.EQ.5) CALL STEER(NCOLS,NROWS,DX,DZ,V)
            IF(CODE.EQ.6) CALL PHASE(NCOLS,NROWS,V)
            CALL SHAPE(V,NSIDE,LO)
            CALL PATTRN(LO,MO,NO,V,DX,DZ)
            STOP
  200       FORMAT(3I0,3F0.0,I0)
  300       FORMAT(8E14.6)
            END
```

REFERENCES

[1] H. L. Nyo and R. F. Harrington, "The Discrete Convolution Method for Solving some Large Moment Matrix Equations," Technical Report No. 21 on Grant No. N00014-76-0025 with the Department of Navy, Office of Naval Research, July 1983.

[2] W. L. Ko and R. Mitra, "A Method for Combining Integral Equation and Asymptotic Techniques for Solving Electromagnetic Scattering Problems," Technical Report 76-6 on Grant No. DAHC04-74-G-0113 with U.S. Army Research Office and Grant No. N00014-75-C-0293 with Office of Naval Research, May 1976.

[3] R. Mittra, Y. Rahmat-Samii, and W. L. Ko, "Spectral Theory of Diffraction," Appl. Phys., vol. 10, pp. 1-13, 1976.

[4] Y. Rahmat-Samii and R. Mittra, "Spectral Analysis of High Frequency Diffraction of an Arbitrary Incident field by a Half Plane -- Comparison with four Asymptotic Techniques," Rad. Sci., vol. 13, pp. 31-48, 1978.

[5] R. Kastner and R. Mittra, "A Spectral-Iteraction Technique for Analyzing Scattering from Arbitrary Bodies, Part I: Conducting Cylinders with E-wave Incidence," IEEE Trans. Antennas and Propagat., vol. AP-31, No. 3, pp. 499-506, May 1983.

[6] A. V. Oppenhiem and R. W. Schafer, "Digital Signal Processing," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[7] R. F. Harrington, "Field Computation by Moment Methods," The Macmillan Company, New York, 1968.

[8] D. C. Kuo and B. J. Strait, "Improved Programs for Analysis of Radiation and Scattering by Configurations of Arbitrarily Bent Thin Wires," Scientific Report No. 15 on Contract No. F19628-68-C-0180 with Air Force Cambridge Research Laboratories, Report AFCRL-72-0051, January 1972.

[9] E. Ngai and A. T. Adams, "A Computer Program for a Planar Array of Thin Wire Dipoles Parallel to a Perfectly Conducting Plane," Scientific Report TR-80-3, Syracuse University, March 1980.